Università degli Studi di Pisa

Facoltà di Scienze Matematiche, Fisiche e Naturali

Corso di Laurea Specialistica in Tecnologie Informatiche

Tesi di Laurea

Formalizzazione di un tipico problema di ACD (Automatic Call Distribution), modellazione e realizzazione della relativa soluzione applicata nell'ambito del Predictive Dialing

Relatore
Prof. Antonio Cisternino

Laureando
Stefano Passatordi

Anno accademico 2007-2008

Alla mia famiglia...

Senza il vostro affetto

Non avrei mai raggiunto

Questo importante traguardo

Grazie!

"Non potrai mai conoscere il tuo limite se non provi a superarlo..."

Indice

Capitolo 4 - Implementazione del modello

Capitolo 5 - Implementazione di un simulatore e confronto pratico tra i modelli

Introduzione

Il primo Call Center[1] è nato nel 1968 negli Stati Uniti. La Ford, famosa casa automobilistica americana, istituì il primo numero verde della storia, per facilitare la comunicazione tra i clienti e l'azienda stessa. Questa iniziativa riscosse un ottimo successo tra la popolazione americana, tanto da indurre tutte le maggiori aziende dell'epoca a seguire l'esempio della Ford.

Per i primi anni, gli operatori dei Call Center hanno offerto ai propri clienti una assitenza del tutto marginale. Non avendo alcun tipo di supporto tecnologico, erano impossibilitati ad offrire un aiuto mirato, ed erano costretti ad attenersi a risposte stardard, in relazione al tipo di quesito pervenuto. Fortunatamente, negli anni '80, con l'avvento di nuove tecnologie, i Call Center diventano sempre più importanti, ed assumono un ruolo centrale nel rapporto con il cliente.

La tecnologica telefonica, applicata alle esigenze delle grandi imprese, immette nel mercato una nuova soluzione: i *distributori automatici di chiamate* (ACD).

Un ACD è un centralino che riceve tutte le chiamate da parte dei clienti e le smista agli operatori liberi. Questa nuova tecnologia è in grado di gestire anche una o più code di chiamate in ingresso. Nel caso che tutti gli operatori siano occupati, l' ACD pone la telefonata del cliente in uno stato di attesa, con priorità in base all'ordine di arrivo della chiamata stessa.

Solo negli anni '90, con la costante crescita della produzione informatica, si sono potuti sviluppare software di CTI (Computer Telephony Integration)[2], che hanno permesso l'integrazione di sistemi telefonici con sistemi software. Grazie a questa sinergia tra diverse tecnologie, l'operatore è stato in grado di integrare le informazioni del cliente, con quelle presenti in un archivio collettivo, garantendo una maggiore efficacia del servizio offerto.

---

[1] Un Call Center è formalmente definito come un ufficio specializzato che riceve e trasmette grosse quantità di richieste tramite telefono.

[2] La Computer Telephony Integration (CTI) è una tecnologia che permette di interfacciare un sistema telefonico con un sistema informatico aziendale.

Con l'avvento dei sistemi CTI, ogni Call Center ha avuto la possibilità di registrare dati riguardanti i propri clienti, potendo così creare un profilo dettagliato per ognuno di essi. Il tempo che le aziende hanno impiegato per capire come sfruttare queste informazioni, è stato minimo. L'attività del Call Center, non rimane mirata solamente al supporto, ma inizia a puntare anche alla vendita di prodotti specifici, in base al profilo del cliente stesso.

Da questo momento in poi, non è solo il cliente che contatta il Call Center, ma avviene anche il contrario. Questo fenomeno dà vita ad un nuovo tipo di Call Center, che effettua solo chiamate in uscita, l'outbound Call Center. Lo scopo di quest' ultimo era, ed è, ancora oggi, il telemarketing. La sua principale attività consite nel contattare quanti più clienti è possibile, per poter proporre la vendita di vari prodotti. Con la diffusione di Internet e delle nuove tecnologie ad esso legate, il Call Center diventa la più importante arma pubblicitaria delle aziende.

Nei primi anni di utilizzo dei Call Center come centri pubblicitari, il grande problema che ogni azienda ha dovuto affrontare, è stato il costo di utilizzo e mantenimento delle linee telefoniche, sulla normale rete pubblica (PSTN)[1]. Non potendo affrontare spese estremamente elevate, ogni Call Center aveva a disposizione un limitato numero di linee telefoniche, in relazione alla sua importanza e potenza economica. Per molti anni, il vincolo imposto dal numero di linee telefoniche a disposizione, ha limitato fortemente il fenomeno degli outbound Call Center.

Ad oggi, la presenza di tecnologie VoIP (Voice Over IP)[2] ha permesso di superare il suddetto limite, provocando un massiccio utilizzo dei Call Center come mezzi pubblicitari.

E' proprio in questa fase, che le persone hanno iniziato a percepire realmente l'esistenza di questo fenomeno, a causa dell'elevato numero di telefonate che partono dai Call Center, per giungere nelle nostre case. Un massiccio, e non controllato, utilizzo delle linee telefoniche, sia PSTN che VoIP, da parte di tutti i Call Center del mondo, ha

---

[1] PSTN è l'acronimo di Public Switched Telephone Network. E' comunemente utilizzato per indicare la rete telefonica pubblica.

[2] Il VoIp è una tecnologia che rende possibile effettuare una conversazione telefonica sfruttando una connessione Internet o un'altra rete dedicata che utilizza il protocollo IP.

indotto alcuni paesi, come l'Inghilterra, ad introdurre nuove norme per limitare l'attività degli outbound Call Center.
Per rispettare i nuovi vincoli legislativi, i moderni Call Center hanno introdotto nuove metodologie per svolgere la loro attività.
In questa ottica, si configura l'utilizzo del Predictive Dialing. Una tecnica che permette, in maniera automatica, di porre in comunicazione cliente ed operatore, cercando di minimizzare il numero delle telefonate effettuate, mantenendo, al contempo, una elevata efficienza. Nonostante sia stato introdotto da oltre quindici anni, il Predictive Dialing rimane un problema tutt'altro che risolto. *[Rif : 1, Call Center, gli schiavi elettronici della new economy - Novembre 2005 - Claudio Cugusi]*

Nei successivi capitoli, il Predictive Dialing verrà affrontato gradualmente. Nel primo capitolo verrà contestualizzato e formalizzato il problema. Nella prima parte, come primo argomento, sarà valutata la struttura di un moderno Call Center. Come sono organizzati e quali sono gli attuali problemi e limiti che caratterizzano un odierno sistema di supporto e telemarketing. Quali sono le tecnologie che vengono utilizzate maggiormente, come i server PBX per effettuare e smistare internamente le telefonate. Verrà introdotto il server PBX Asterisk, un prodotto open source, tra i più diffusi nel settore. Nella seconda parte del primo capitolo, verrà introdotto e contestualizzato il problema del Predictive Dialing. Come è nato e perché è un fattore critico per tutti i Call Center. Infine, verrà data una sua interpretazione formale per inquadrare il problema da un punto di vista matematico.
Il secondo capitolo introduce e spiega quali sono i modelli risolutivi attualmente utilizzati. La letteratura in merito è molto scarsa, esiste un solo articolo che tratta il Predictive Dialing, dandone anche una soluzione. E' un saggio di Samuelson, imprenditore e professore, nonché inventore del Predictive Dialing stesso.
Sono due le soluzioni note in letteratura, una spiegata nell'articolo di cui sopra, ed un'altra che è presente in tutti gli scritti riguardanti questo problema, la soluzione basata sul fattore "ratio".
Dopo aver introdotto e spiegato le due soluzioni attualmente note, nel terzo capitolo, verrà proposta una nuova metodologia risolutiva, con lo scopo di rispondere in maniera efficiente al Predictive Dialing. Il modello proposto ha una prima rappresentazione

astratta, che lo rende indipendente da eventuali particolari implementazioni. Successivamente, verrà data una possibile implementazione del modello astratto, con annesso anche un esempio pratico di utilizzo. L'ultima parte del terzo capitolo, è dedicata ad un confronto tra i modelli esistenti in letteratura ed il modello proposto:
quali sono i vantaggi e gli svantaggi di ogni soluzione e quale è quella che potrebbe risultare la più efficiente.
Il quarto capitolo descrive come è stato implementato il modello proposto, quali sono state le tecnologie utilizzate e le motivazioni di tali scelte. Verranno descritti i principali punti dell'algoritmo implementato e come si è arrivati ad una tale soluzione.
Nell'ultimo capitolo, il quinto, sarà discusso il simulatore per outbound Call Center che è stato implementato per poter confrontare tutti modelli da un punto di vista pratico, come è stato pensato il suo funzionamento e la sua struttura, oltre al perché di alcune scelte importanti, per rendere la simulazione affidabile. L'ultima parte del quinto capitolo, è dedicata ai risultati delle simulazioni ed alle considerazioni che è possibile effettuare in base ai valori ottenuti.

Capitolo 1

Call Center e Predictive Dialing

1.1 Struttura di un moderno Call Center

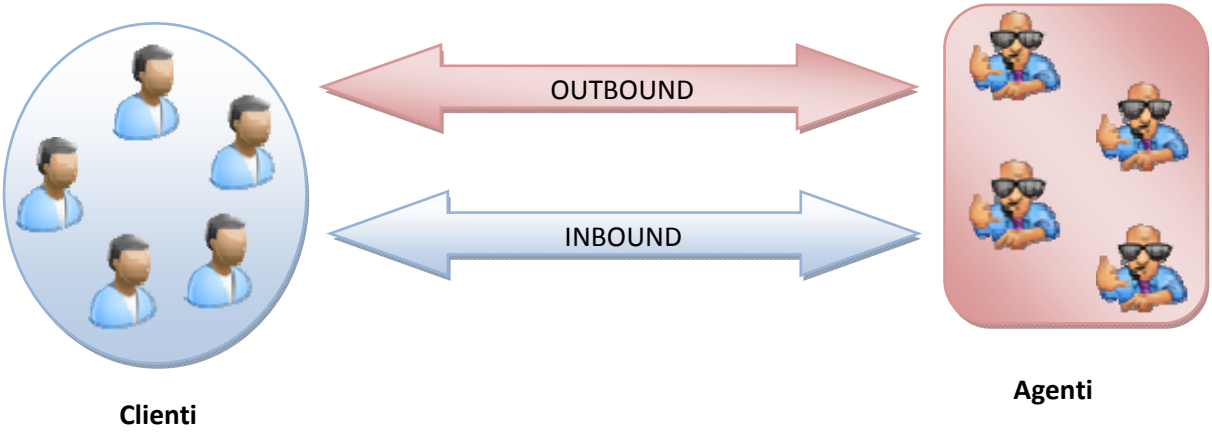
Negli ultimi anni, i Call Center si sono affermati, sempre più, come un ottimo mezzo a disposizione delle aziende per vendere i propri prodotti/serivzi e per offrire assistenza ai propri clienti.

La loro diffusione è in continua crescita, grazie anche allo sviluppo delle tecnologie VoIP. Sempre molte più realtà industriali tendono ad esternalizzare tale servizio ad altre aziende, che impiegano lavoratori e tecnologie specializzate per adempiere al meglio alle richieste del cliente.

Grazie ad Internet, i Call Center hanno ampliato il loro spettro funzionale e sono diventati Contact Center. La differenza consiste nelle diverse modalità di comunicazione tra operatore e cliente. In un Call Center l'interazione avviene solo tramite telefono. In un Contact Center, invece, oltre al telefono, vengono utilizzati fax, email ed sms.

In entrambi i casi, il modello di funzionamento è sempre lo stesso. L'interazione possibile è a doppio senso. Gli agenti generano del traffico outbound, ovvero in uscita dal sistema, per contattare i clienti che, a loro volta, possono contattare un agente generando del traffico inbound, ovvero in ingresso al sistema. *[Rif : 2, An Overview of Routing and Staffing Algorithms in Multi-Skill Customer Contact Centers – Marzo 2006 - Ger Koole & Auke Pot]*

Di seguito, è riportata una schematizzazione dei due tipi di traffico che caratterizzano tutti i sistemi di telemarketing.



(Figura 1)

Gli obiettivi di un Call Center, più in generale, di un Contact Center sono essenzialmente due :

1. offrire una pronta ed efficiente assistenza ai clienti
2. vendere i propri prodotti/servizi

In entrambi i casi, il numero di agenti è limitato e dipende da quanto è ricca ed importante l'azienda, ma, soprattuto, da quanto l'azienda decida di investire su questo tipo di servizio.

Offrire assistenza ai clienti genera il traffico *inbound* al Call Center. In quest'ultimo caso, l'obiettivo dell'azienda è quello di minimizzare, a vantaggio del cliente, i tempi di attesa prima che l'operatore risponda. In questa circostanza, la minimizzazione del tempo di attesa si configura come un problema di teoria delle code. Il sistema viene modellato come una coda M/M/n, con una distribuzione degli arrivi che è quella discreta di Poisson. Per stimare la probabilità dei tempi di attesa in coda viene utilizzata la variabile casuale Erlang C. *[Rif : 3, Markov chain models of a telephone call center with call blending – Alexandre Deslauriers,Pierre L'Ecuyer, Juta Pichitlamken,Armann Ingolfsson, Athanassios N.Avramidis]*

In questo caso, il problema va risolto studiando la coda delle rischieste in ingresso, cercando di ottimizzare lo smistamento delle stesse, in base alle risorse attualmente a disposizione.

La vendita da parte degli operatori genera, invece, il traffico *outbound* dal Call Center. In questo caso, l'obiettivo dell'azienda è quello di vendere il più possibile con le risorse umane a disposizione.
Indipendentemente dalla bravura degli operatori, dal Call Center deve essere effettuato un certo numero di chiamate, che garantisca all'azienda un volume minimo di vendita ogni giorno lavorativo. Fino agli anni '90, gli operatori dei Call Center chiamavano, uno ad uno, i numeri a loro assegnati per le vendite. Schedulavano in maniera autonoma il loro lavoro, sotto gli attenti sguardi degli ispettori, il cui compito era quello di assicursi che tutti lavorassero senza troppe pause inutili.
Con questo meccanismo, l'inefficienza degli outbound Call Center era altissima. Sia perché spesso gli operatori fingevano di parlare, sia perché la maggior parte del tempo la spendevano per contattare numeri occupati o fuori servizio.
Sia nel caso dell'assistenza che nel caso delle vendite, gli agenti possono essere suddivisi in due categorie, agente *single-skill* e *multi-skill*. Nel primo caso, un agente è specializzato solo in un tipo di assistenza o campagna vendite. Nel secondo caso, l'operatore può essere specializzato in più ambiti dell'assistenza, così come in diverse campagne promozionali. La scelta di avere un Call Center di tipo single-skill o multi-skill è molto critica, ed ha conseguenze importanti sia sulla struttura del servizio, che sugli algoritmi da utilizzare per rendere il Call Center operativo. *[Rif : 2]*
Verso la fine degli anni '90, i Call Center vengono muniti di sistemi hardware e software, che hanno risolto tanti dei problemi sopra citati.
Da anni, i ricercatori del settore lavorano per individuare una soluzione, flessibile e non vincolante rispetto ad una sola campagna pubblicitaria, e che ottimizzi il tempo di chiamata degli operatori. Lo scopo è quello di porli in condizione di effettuare sempre chiamate utili per le vendite.

1.1.1 Private Branch eXchange (PBX)

Il cuore di tutti i moderni Call Canter è il PBX (Private Branch Exchange). Un insieme di componenti hardware e moduli software che, in pratica, svolgono automaticamente tutte le attività che prima gli operatori eseguivano a mano. Il PBX ha svariati compiti, tra cui, sicuramente, quello di mettere in comunicazione tra loro numeri interni ad una azienda, e di permettere anche le comunicazioni da e verso l'esterno.
I modelli di ultima generazione consentono comunicazioni sia su normale rete telefonica PSTN, che tramite protocolli che sfruttano internet, come quelli della famiglia VoIP. [Rif : 4, *PBX Systems for IP Telephony - Cap. 1 – Aprile 2002- Allan Sulkin*]

Un PBX ha numerosissime funzionalità, tra le più importanti si ricordano :

- Call Accounting: un registro di log per tutte le attività del PBX stesso. Tra le principali voci, ci sono il numero di chiamate effettuate in ingresso ed in uscita, il numero di squilli ed il numero di linee occupate e fuori servizio.

- Call Waiting: mette in pausa una telefonata, se una delle due parti non è ancora pronta per iniziare la comunicazione.

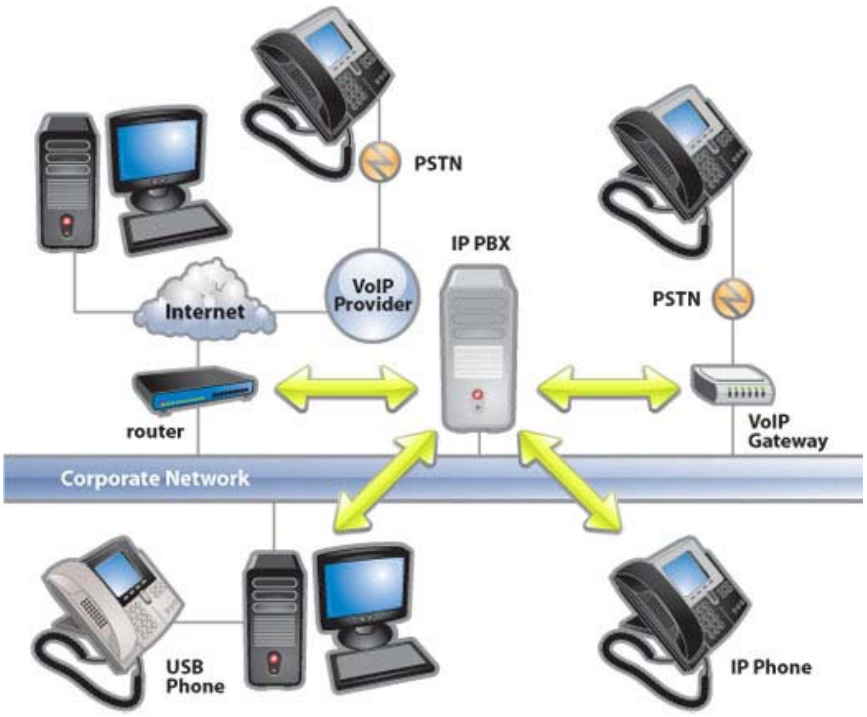
- Conference Call: permette che più parti siano coinvolte contemporaneamente in una conversazione nello stesso istante.

- Voice Mail: registra messaggi vocali, per poterli inviare a più destinatari contemporaneamente.

- Interactive Voice Response (IVR): permette ad un utente di interagire, tramite la digitazione dei tasti telefonici, con una voce registrata che lo guida tra varie scelte, per individuare il motivo della chiamata e comportarsi di conseguenza.

- Deviazione chiamata: permette di deviare una chiamata in entrata, in base a varie condizioni (identificativo chiamante, orario) su un altro interno, un gruppo, sulla casella vocale, su un numero esterno.

- Automatic Call Distribution (ACD): in maniera automatica vengono smistate le chiamate in ingresso verso un particolare gruppo di agenti, in base al tipo di chiamata ricevuta.



(Figura 2)

La Figura 2 rende bene l'idea del ruolo di un server PBX: rappresenta un punto di centralizzazione per tutte le comunicazione sia in ingresso che in uscita, capace di interfacciarsi simultaneamente con diverse tecnologie, fornendo omogeneità al sistema che lo ospita.

1.2 Asterisk

Attualmente esistono varie soluzioni proprietarie che implementano un PBX, le più famose sono sicuramente quelle della Oracle e dell'IBM.

Come prodotto open source esiste il diffusissimo PBX Asterisk, un progetto scritto nel linguaggio C, gestito dalla Digium[1], totalmente gratutito (licenza GNU General Public License) e con un buon supporto.

Asterisk è la soluzione di tantissime aziende a livello mondiale perché, a costi ridotti, si può avere un PBX completo e molto flessibile. Può essere utilizzato sia in ambito privato, come normale centralino, che in grosse realtà aziendali, come Call Center sia per il traffico inbound che outbound.

Per capire bene cosa sia Asterisk e quali sono le sue potenzialità è possibile fare degli esempi concreti di suoi possibili utilizzi.

E' una piattaforma molto potente e con un pò di fantasia è possibile utilizzarla per automatizzare azioni che non riguardano il mondo dei Call Center.

Oltre agli utilizzi precedentemenți elencati, è possibile usarlo, ad esempio, come una segreteria telefonica intelligente. Registra il messaggio vocale e successivamente può comporre il numero del proprietario e fargli ascoltare il messaggio.

Lo stesso servizio può essere eseguito anche sulla ricezione di nuove email o fax.

L'importante è che sia fisicamente collegato ai vari dispositivi che monitorizza ed abbia un accesso alla linea telefonica. Un altro suo interessante utilizzo è la multiconferenza.

Più persone, sicuramente più di due, che si trovano fisicamente in posti diversi, possono comporre un numero telefonico, monitorato da Asterisk, e grazie ai dialplan (spiegati in seguito), possono essere messi in comunicazione tra di loro e parlare come se fossero vicini. Nel campo della sicurezza domestica, può essere collegato a sistemi di allarme o anti incendio. Se si verifica qualche evento spiacevole, il proprietario di casa viene chiamato sul cellulare ed un messaggio registrato, in base all'evento, può avvisare sulla situazione nell'appartamento.

---

[1] La Digium, Inc. è una azienda che ha sede in Alabama e si occupa essenzialemente di sviluppare hardware per PBX

Insomma i suoi utilizzi sono molteplici e riguardano vari ambiti, non sono solo limitati alle attività di supporto o telemarketing dei Call Center.
La sua flessibilità è garantita dal fatto che è possibile accedere, non solo al codice, ma anche a tutti i suoi file di configurazione in maniera del tutto immediata.

Inoltre, espone delle API appositamente pensate per interagire con sistemi eterogenei e distribuiti (vedi in seguito).
Asterisk gira su sistemi GNU/Linux per x/86 ed offre tutte le funzionalità che possono essere richieste ad un PBX.
Ha una comoda, ma non stabilissima, interfaccia web (vedi in seguito e cap. 4) e supporta numerosi protocolli VoIP :

- IAX (Inter-Asterisk Exchange)
- H.323
- SIP (Session Initiation Protocol)
- MGCP (Media Gateway Control Protocol)
- SCCP (Cisco Skinny)

Al suo avvio, Asterisk legge dal file di configurazione (*modules.conf*) quali sono i moduli da caricare, ovvero quali saranno i servizi richiesti durante la sua fase di servizio. Questo meccanismo è estremamente comodo, è inutile caricare ed appesantire il sistema con moduli che non saranon mai usati in fase di servizio.

Le pricipali entità su cui si basa il core di Asterisk sono :

- canali di comunicazione
- utenti
- agenti
- dialplan

I *canali di comunicazione* rappresentano il tipo di interfaccia attraverso cui si può comunicare. Ad esempio, l'interfaccia Ethernet viene sfruttata dai canali SIP o H.323, ovvero della famiglia VoIP.
In appositi file di configurazione, presenti nella directory */etc/asterisk*, è possibile creare e configurare i canali che saranno sfruttati durante le comunicazioni.
Gli *utenti* sono strettamente legati al protocollo, non riguardano persone fisiche, ma solo utenze virtuali di tipo SIP, IAX, eccetera. Ogni utente ha delle caratteristiche ben precise, che dipendono dal tipo di protocollo a cui sono legati. Ad esempio, un utente IAX può richiedere una connessione autenticata e cifrata non supportata dall'utente SIP.

L'*agente* rappresenta la persona fisica che utilizza il sistema da una qualsiasi postazione, a cui è associato un preciso utente. In pratica l'utente è assegnato ad ogni postazione in base al protocollo usato, mentre l'agente è la rappresentazione della persona fisica, che può cambiare postazione e quindi utente. Utente e agente rappresentano due diversi livelli di astrazione, e sicuramente l'agente è il livello più alto. Chiamando un certo utente, squillerà sempre e solo il telefono ad esso associato. Mentre, chiamando un agente, squillerà il telefono da cui l'agente stesso ha effettuato il login nel sistema.

Il *dialplan* è il fulcro di tutta la piattaforma. Definisce come vengono gestite le chiamate in ingresso ed in uscita dal sistema. Questa gestione è molto potente, nel dialplan è possibile associare, ad ogni chiamata, una serie di istruzioni da eseguire, in modo simile a un linguaggio di scripting.

E' contenuto nel file *extensions.conf* ed è composto principalmente da quattro elementi:

- contesti
- estensioni
- applicazioni
- code di chiamata

I *contesti* servono solo a dividere logicamente i vari insiemi di istruzioni e comportamenti del sistema. Asterisk sceglie il contesto in base alla configurazione dei vari utenti o linee tradizionali. Ogni chiamata, da una particolare utenza, verrà gestita dalle istruzioni del relativo contesto.

Le *estensioni* sono delle azioni da eseguire in risposta alla chiamata ad un determinato numero composto. In questo modo, tramite la digitazione di tasti sul telefono, è possibile avviare l'esecuzione di particolari azioni, come, ad esempio, il login dell'agente nel sistema.

Le *applicazioni* sono dei blocchi di codice già pronti e disponibili, che possono essere composti tra loro per eseguire azioni complicate in maniera automatica e veloce.

Le *code di chiamata* sono la funzionalità più importante quando si vuole usare Asterisk come Call Center. Permettono di organizzare gli agenti in code logiche, di gestire le chiamate in ingresso, mettendole in attesa o passandole direttamente ad un operatore libero. Offrono la possibilità di far ascoltare all'utente degli annunci registrati o della semplice musica. Durante una campagna di telemarketing, se vengono eseguite promozioni su prodotti diversi, le code sono lo strumento per dividere logicamente gli operatori secondo i prodotti stessi. Ad esempio, la coda 1 servirà per il prodotto A, la coda 2 per il prodotto B e così anche per gli eventuali altri prodotti.

La sintassi utilizzata per i dial plan è la seguente :

```
[context]
  exten => id, priority, command
```

Si distinguono chiaramente, il contesto (context), l'estensione (indicata dalla parola chiave exten) che è l'intera riga *exten => id, priority, command* e le applicazioni (command). L'Id è un identificativo alfanumerico per indicare a quale estensione si fa riferimento. La priorità (priority) serve a dare un ordine nell'esecuzione delle estensioni

dello stesso contesto. A parità di priorità, viene eseguita la prima che viene incontrata dall'alto verso il basso.
Ad esempio, il seguente codice serve per effettuare il login di un agente nel sistema, usando il telefono :

```
1. [internal]
2. exten => 701,1,VMAuthenticate(@agent|)
3. exten =>
   701,2,AddQueueMember(SALES|local/${AUTH_MAILBOX}@agent
   s/n)
4. exten =>
   701,3,AddQueueMember(SUPPORT|local/${AUTH_MAILBOX}@age
   nts/n)
5. exten => 701,6,Playback(agent-loginok)
6. exten => 701,7,Playback(goodbye)
7. exten => 701,8,Hangup
```

Nella prima riga troviamo il contesto, internal, che serve solo ad identificare univocamente questo constesto dagli altri del sistema.
Dalla riga 2 alla 7 ci sono una serie di estensioni, introdotte dalla parola chiave exten.
Il significato della prima estensione, riga 2, è che digitando sul telefono il numero 701, viene richiamata una applicazione, VMAuthenticate, che è già presente in Asterisk di default. Il compito di questa macro è quello di far partire una voce registrata che dice all'agente di digitare, sul telefono, prima il suo identificativo e successivamente la sua password. Al termine l'utente sarà loggato nel sistema. Viene, poi, eseguita la seconda estensione, riga 3, che aggiunge l'operatore appena loggato nella coda indicata con Sales e poi fa lo stesso,riga 4, per la coda Support.
Il compito delle estensioni nella riga 5 e 6 è semplicemente quello di far ascoltare all'operatore un messaggio di successo per il login e di saluto.
L'ultima azione, riga 7, chiude la chiamata e da quel momento in poi, l'agente è disponibile per le due Sales e Support.

Per poter interagire tramite tasti telefonici, sia l'identificativo dell'agente che la sua password, devono essere necessariamente dei numeri interi.
Così come questo esempio, con lo stesso meccanismo è possibile eseguire una miriade di azioni per interagire con il cuore del sistema.
Inoltre, grazie alle API (Application Programming Interface) esposte da Asterisk, è possibile intercettare la catena degli eventi, scatenati dalle code, e catturare tutte le informazioni di una chiamata sia in ingresso che in uscita.
E' possibile interagire con queste API con i linguaggi di programmazione più diffusi : Perl, PHP, C, Pascal, Bourne Shell, C++, C# ed altri. *[Rif : 5, Asterisk: The Future of Telephony - O'Reilly, 2007 - Jim Van Meggelen, Leif Madsen, Jared Smith]*

Le API sono divise in due grosse famiglie :
- AGI (Fast Asterisk Gateway Interface)
- Manager

Le AGI, offrono la possibilità di controllare i dial plan contenuti nel file extensions.conf. In realtà il gruppo AGI è a sua volta suddiviso in :

- EAGI : offre la possibilità di gestire il canale audio per farlo interagire con i vari dialplan.
- FastAGI : offre la possibilità di inviare comandi AGI ad un server remoto attraverso la rete.
- DeadAGI : offre la possibilitò di interagire con un canale non più in uso, dopo che una chiamata viene terminata.

Le Manager API permettono, ad un client, di potersi connettere e poter comunicare con un server Asterisk sfruttando il protocollo TCP/IP.
In questa sede, verrano introdotte solo le Manager API, poiché sono quelle che realmente sono servite durante l'implementazione del modello proposto (vedi Cap. 4).
Per poter comunicare ed interagire con il server da remoto, le Manager API, sono l'unica possibilità offerta da Asterisk. Il loro utilizzo, permette, ad un client, di poter gestire tutti gli eventi e le azioni, sia riguardanti le chiamate che gli agenti stessi. Il protocollo di comunicazione tra client e server, consiste in una stringa composta da

"chiave: valore", terminante con un CRLF (Carriage Return Line Feed). Questa stringa, così composta, sarà, da ora in poi, riferita come *pacchetto*.

Prima di poter catturare gli eventi o inviare azioni al server, il client deve aprire una sessione, sfruttando l'API *Login* o *AgentCallBackLogin*. Entrambe permettono, ad un client, di inviare al server le proprie credenziali ed essere, quindi, accettati dal sistema, se si hanno i giusti diritti. La differenza tra le due chiamate, consiste nel fatto che Login stabilisce una connessione sempre aperta con il server, mentre, l'altra, apre e chiude le connessioni solo quando necessario.
Dopo l'autenticazione, i pacchetti posso essere invitati dal client al server, e viceversa, in qualsiasi momento. L'unica differenza consiste nella tipologia di pacchetto inviato.
Se è il client ad inviare, allora la chiave sarà sempre *Action.* Se, invece, ad inviare è il server, la chiave potrà essere *Event* o *Response,* in base a ciò che il client ha richiesto.
Il terminatore CRLF, viene utilizzato, sia dal client che dal server, per capire che il pacchetto è terminato e può essere processato.
In base a quanto scritto in precedenza, ci sono tre tipi possibili di pacchetti :

- Action
- Event
- Response

Le azioni vengono inviate dal client al server, per indicare il tipo di servizio richiesto. Il valore della chiave, contiene il nome dell'azione richiesta ed i relativi parametri. Le possibili azioni a disposizione, sono circa 50, servono per effettuare operazioni come :

- Login e Logoff dal sistema
- Inizio e terminazione di una telefonata
- Aggiunta o rimozione di elementi da una coda
- Ottenere informazioni di stato dal server
- Settare alcune configurazioni del server
- Inserire o rimuovere una telefonata dallo stato di pausa
- Ottenere informazioni sullo stato degli agenti e delle linee

- Impostare lo stato di pausa o meno degli agenti
- Gestire le conferenze
- Gestire i redirect delle chiamate
- Leggere e scrivere nei file di log del sistema
- Avviare un conferenza e gestirne la moderazione

Gli eventi vengono scatenati dal server e vengono inviati in pacchetti, il cui valore contiene il nome dell'evento ed i relativi argomenti. Sono più del doppio delle azioni, e servono a monitorare, ad esempio :

- Le azioni ed i cambiamenti di stato degli agenti
- I cambiamenti di stato dei canali
- Le azioni ed i cambiamenti di stato delle code
- Lo stato delle chiamate, dalla composizione del numero alla sua terminazione
- Lo stato delle conferenze

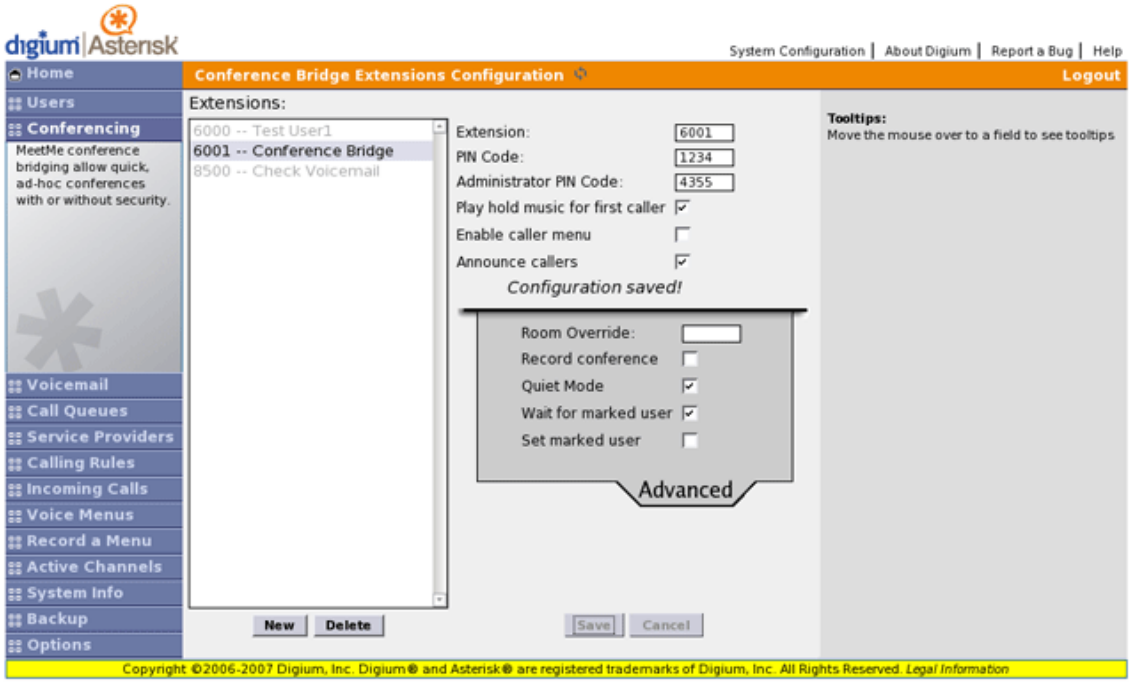
Tramite gli eventi è possibile avere un pieno e totale controllo su quello che sta avvenendo nel server, in qualsiasi istante.
Le risposte vengono inviate dal server al client, successivamente ad una richiesta di esecuzione di una azione. Il loro valore contiene solo l'esito dell'esecuzione dell'azione, successo o errore. In quest'ultimo caso, viene inviato anche il tipo di errore e la causa.
Poiché ogni azione può ricevere una risposta in maniera asincrona, ad ognuna di esse viene associato un identificatore univoco che è presente anche nella risposta. In questo modo è sempre possibile verificare a quale azione si riferisce la risposta ricevuta dal server.

Grazie all'interfaccia web, è possibile loggarsi come amministratore del sistema e gestire, ad esempio:

- Agenti
- Code
- Dial plan
- Protocolli da utilizzare

La GUI (Graphical User Interface) permette di poter controllare, comodamente e velocemente, tutti gli aspetti del server PBX, dai file di configurazione agli agenti, dalle campagne pubblicitarie ai dispositivi collegati al server.

Di seguito una immagine di esempio dell'interfaccia di Asterisk:



(Figura 3)

E' fondamentale spiegare che con Asterisk, ogni campagna pubblicitaria viene associata ad una coda. Ad esempio, è possibile avere un coda chiamata "Pasta", che verrà utilizzata per fare un certo tipo di pubblicità relativo ad un prodotto specifico.
Tutti gli agenti che vengono assegnati, tramite la GUI, alla suddetta coda, hanno il compito di pubblicizzare solo il prodotto relativo alla coda.
Lo stesso vale per una coda chiamata "Automobile". Sarebbe la pubblicità di una automobile e tutti gli agenti appartenenti a quella coda devono pubblicizzare quel prodotto. E' possibile che uno stesso agenta venga assegnato a più code, in quel caso sarà lui a decidere quale pubblicità eseguire per ogni chiamata.

Questa è solo una brevissima panoramica su Asterisk, in realtà questo software è molto più complesso ed ampio, ma quanto detto sarà sufficiente per capire i successivi capitoli.

1.3 Predictive Dialing

Nei primi anni ’80, solo pochissime aziende americane avevano iniziato ad utilizzare i Call Center per effettuare campagne pubblicitarie. Sin dai primi mesi di lavoro, i responsabili dei Call Center notarono che, nonostante le tante ore lavorative, i loro operatori avevano un basso profitto di vendita. Solo dopo attente valutazioni, capirono che tutto dipendeva dal fatto che gli agenti fossero costretti a comporre manualmente i vari numeri. Questo comportava una enorme perdita di tempo e, quindi, tante meno vendite. Senza considerare il tempo medio per comporre un numero telefonico, spesso gli operatori sbagliavano qualche cifra ed erano costretti a ricominciare la telefonata. Quando componevano il numero corretto, esso poteva risultare fuori servizio, un fax oppure occupato. La composizione manuale era evidentemente una pessima metodologia.
Secondo alcuni dati riportati in un comunicato stampa dell’azienda Altesys *[Rif : 6]*, nell’arco di un’ora lavoriativa, solo poco più di 17 minuti erano realmente sfruttati per parlare con un cliente. I circa 43 minuti restanti venivano spesi in composizione dei numeri, errori umani ed in telefonate inutili.
Nacque, quindi, l’esigenza di risolvere questo problema, sviluppando sistemi hardware e software per la composizione automatica dei numeri. I primi *autodialer* furono introdotti all’inizio degli anni ’80 ed avevano il mero compito di comporre automaticamente i numeri, esonerando l’operatore da tale onere. Erano sistemi senza intelligenza, che, partendo da una lista di recapiti telefonici, eseguivano un numero di telefonate pari al numero degli agenti disponibili al momento della composizione. In pratica, questi sistemi automatici, facevano risparmiare solamenente il tempo di composizione del numero, oltre a quello perso a causa degli errori umani. Non risolvevano, però, i ritardi dovuti alle telefonate che risultavano senza risposta, occupate o fax.

Sempre secondo i dati riportati dall' Altesys, gli autodialer hanno incrementato i minuti utili di conversazione fino ad arrivare a circa 24 nell'arco di un'ora.
Questo risultato non era ancora soddisfacente per le aziende del settore, le quali pochi anni dopo, nei primi anni '90, introdussero una nuova tecnologia, il *Predictive Dialer*. Quest'ultimo è un autodialer dotato di intelligenza, capace di predire l'esito e la durata delle telefonate. Sfruttando complessi algoritmi, il sistema predice il numero di agenti disponibili, per ogni istante *t*, ed esegue un certo numero di telefonate, cercando di garantire che tutti gli operatori liberi vengano messi in comunicazione con un cliente diverso. Grazie a questo meccanismo automatico, sia il cliente che l'operatore sentono squillare il telefono e devono semplicemente rispondere, senza dover eseguire altre operazioni.

I moderni PBX sono dotati di un modulo apposito per il Predictive Dialing, anche Asterisk ne ha uno. Non tutti però sono efficienti, ed in realtà sono degli autodialer leggermente più intelligenti. Piuttosto che avere un comportamente predittivo, tendono ad effettuare un numero di chiamate proporzionale agli agenti disponibili.

Un modulo *ideale* per il Predictive Dialing:

1. accede ad una sorgente dati in cui sono memorizzati tutti i numeri dei potenziali clienti, tipicamente interagisce con un CRM (Customer relationship management, un sistema complesso che tiene traccia di tutte le informazioni riguardanti il cliente e lo storico dei contatti avuti con lo stesso.)

2. predice il numero degli agenti liberi in un istante t

3. effettua un ben determinato numero di telefonate, negli istanti subito precedenti a t, e ne controlla l'esito. Quando un numero è inattivo o è un FAX allora lo inserisce in una *black-list* (lista logica che raggruppa i numeri da non usare in futuro), e non viene più utilizzato. Quando è occupato lo inserisce in una *busy-list* (come la precedente lista, ma questa volta i numeri vengono ricontattati in seguito) e verrà ricontattato dopo un certo lasso di tempo. Se il numero chiamato risulta libero, ma nessuno risponde, allora conclude la telefonata. Per capire che

un cliente ha risposto, il sistema può sfruttare, anche, tecnologie a riconoscimento vocale. In questo caso, il modulo cerca il primo agente disponibile e mette in contatto il numero chiamato con l'operatore scelto.

In sostanza, il Predictive Dialer *ideale* elimina tutti i tempi morti dell'operatore e lo pone sempre e costantemente nella situazione di poter parlare con un cliente.
Questa è, però, una visione ottimistica e poco reale. Tutti gli attuali moduli di Predictive Dialing, presenti sul mercato, non riescono a garantire l'efficienza del modello ideale.
Tutte le aziende produttrici di questo tipo di tecnologia, come l' Altesys, promettono un incremento dei tempi di conversazione per agente, pari al 70% rispetto al sistema manuale.

	Modalità di funzionamento:			
	Manuale	**Preview**	**Predittivo**	**Predittivo**
Numero Agenti			15 Agenti	50 Agenti
% di chiamate perse			1% Abbandono	5% Abbandono
% Tempo in conversazione	29%	39%	**70%**	**76%**
Tempo in conversazione su 60 minuti di lavoro	17:24	23:24	**41:59**	**45:27**

(Figura 4)

Il comportamento predittivo, si verifica quando il modulo effettua un certo numero di telefonate, calcolato predicendo il numero di agenti liberi quando i clienti risponderanno. La predizione avrà successo se, tra tutte le telefonate effettuate, quelle risposte dai clienti saranno pari al numero di tutti gli agenti liberi. Se la predizione fallisce, si verifica che uno o più agenti non lavora, oppure che sono tutti occupati ma ci sono ancora chiamate risposte dai clienti. In quest'ultimo caso, le telefonate saranno concluse automaticamente dal sistema. Questo tipo di comportamento, ha come effetto collaterale la generazione delle *silent call*.
Una silent call si verifica quando il Predictive Dialer fallisce la predizione ed effettua più telefonate rispetto agli agenti realmente disponibili. Questo provoca delle chiamate

dette, appunto, silenziose. Il cliente risponde ma non c'è nessun agente dall'altro lato che fa lo stesso. I moderni algoritmi per il predictive dialing tendono molto a favorire l'effetto collaterale delle silent call, piuttosto che lasciare uno o più operatori senza una telefonata da servire. Per questo motivo, sia in Inghilterra che negli Stati Uniti, sono state introdotte delle leggi per limitare il numero di silent call, diventato troppo elevato.
Non più del 3% del totale delle telefonate effettuate deve risultare in una silent call, pena il pagamento di una multa. Di conseguenza, i moderni algoritmi devono tenere in considerazione anche questo aspetto. *[Rif : 2]*

Esistono principalemte quattro tipi di Predictive Dialer :

1. Software dialer
2. Hardware dialer
3. Smart dialer
4. Mixed dialer

La soluzione basata solo sul software è quella più economica, poiché non prevede hardware aggiuntivo. Interagisce con il PBX tramite collegamenti CTI, ed esegue solo l'algoritmo predittivo. Non avendo componenti hardware non può offrire servizi quali: registrazione vocale, riconoscimento vocale ed IVR. E' una buona soluzione in ambienti distribuiti perché è molto flessibile, ma non è molto efficiente nella valutazione degli esiti delle telefonate.
La seconda soluzione, basata sull'hardware, è molto più costosa della prima perché si basa sull'utilizzo di particolari componenti hardware dedicati. E' completamente indipendente dal tipo di PBX, e si collega ad esso tramite una normale connessione telefonica. Potendo sfruttare componenti hardware dedicati, permette non solo l'esecuzione efficiente dell'algoritmo predittivo, ma espone anche altri servizi. Tra i principali, ad esempio, la possibilità di effettuare le telefonate sia in ingresso che in uscita senza passare dal PBX, la possibilità di creare ed aggiornare un database con tutti gli eventi gestiti dal modulo stesso (per fini statistici e di controllo) e la possibilità di tener traccia dell'esito della vendita, relativamente ad un particolare operatore.

Tutte queste allettanti caratterische hanno, però, un elevato costo sia di acquisto che di mantenimento. Inoltre, si consideri che alcune sue funzionalità sono già presenti nei PBX attualmente a disposizione e risultano, quindi, ridondanti.
Gli Smart dialer sono degli hard-dialer più snelli dal punto di vista dei servizi offerti e meno costosi, sia per l'acquisto che per la manutenzione.
Si differenziano da tutti gli altri tipi di predictive dialer perché, prima dell'agente, il cliente interagisce con una voce registrata.
Questa registrazione ha il compito di spiegare al cliente il motivo della chiamata. La telefonata viene smistata ad un agente, solo se l'utente è realmente interessato.
Sicuramente questo meccanismo permette di avere meno agenti a disposizione, poiché le chiamate vengono smistate solo in caso di reale interesse.
Allo stesso tempo, però, fa diminuire il numero di potenziali clienti perché molte persone terminano la telefonata in anticipo, quando iniziano ad ascoltare il messaggio registrato. La soluzione mixed è basata su componenti hardware, non per forza costosi, e su un potente strato software che offre flessibilità ed estendibilità funzionale.
Quest'ultima, è sicuramente la soluzione maggiormente diffusa negli outbound Call Center attuali.

Indipendentemente dal tipo di soluzione software e/o hardware adottata, il predictive dialing rimane un problema ancora aperto e molto complesso, poiché le variabili in gioco sono numerose. Il punto di forza di un buon algoritmo predittivo è sicuramente l'insieme di informazioni su cui effettuare la predizione. Una valida scelta di queste informazioni può fare la differenza rispetto ad algoritmi concorrenti.

1.4 Formalizzazione del problema

La definiziona formale del problema legato al predictive dialing è la seguente.
Siano:
- t_s un qualsiasi valore dell'asse temporale
- *j* il numero massimo di linee telefoniche a disposizione
- *Op* il numero di agenti realmente a disposizione nell'istante t_s
- *q* il tempo medio per comporre un numero telefonico
- *S* l'insieme delle informazioni scelte

Dato t_s, predire, partendo dall'insieme di informazioni S,qual è il numero K, con K<= j, di telefonate da effettuare nell'istante $r_s = t_s - q$, affinché a tutti gli operatori liberi Op venga smistata una telefonata.

Questa formalizzazione non tiene conto del problema delle silent call. In realtà, con il tempo, tutti gli outbound Call Center dovranno adeguarsi alle nuove normative che i vari governi di tutto il mondo stanno approvando.

Il numero K, deve, quindi, essere anche il più basso tra quelli che, comunque, soddisfano la condizione di mantenere tutti gli operatori impegnati.
[Rif : 7, Using simulation to predict market behavior for outbound call centers - 2007 - Paulo J. de Freitas Filho, Geovani Ferreira da Cruz, Rui Seara, Guilherme Steinmann]

Capitolo 2

Ricerca e valutazione di modelli esistenti

2.1 Letteratura esistente

Il Predictive Dialing è un argomento di nicchia, che interessa esclusivamente gli outbound Call Center. Sono poche le aziende che hanno investito tempo e denaro per risolvere questo problema e, in ogni caso, non hanno mai reso pubblici i loro risultati. Il motivo di questa scelta è sicuramente legato al fattore economico. I primi algoritmi di Predictive Dialing, sviluppati all'inizio degli anni '90, hanno fruttato alle aziende milioni di dollari. I piccoli Call Center, non potendo investire né tempo e né denaro in ricerca, preferivano acquistare direttamente la soluzione del problema da altre aziende che, invece, avevano intuito l'importanza economica di un tale servizio.

I normali inbound Call Center, sono stati studiati sin dai primi anni '70, ed ancora oggi si studia come predire l'arrivo delle chiamate. Il motivo è ovvio, è difficile far funzionare un inbound Call Center, se non si ha almeno una idea approssimativa di quanti operatori avere costantemente a disposizione. Questo discorso, però, non è mai valso per gli outbound Call Center. Avendo a disposizione un qualsiasi numero di agenti, è sempre possibile effettuare un numero di telefonate pari almeno al numero degli agenti stessi.

Per i suddetti motivi, la letteratura esistente su questo argomento, è praticamente quasi nulla.

Sicuramente di grande importanza nel settore, è l'articolo scritto da Samuelson[1], che si definisce l'inventore del Predictive Dialing.

Egli ha dato vita la primo modello, sia teorico che concreto, di Predictive Dialing e per molti anni è stato anche l'unico a disposizione delle aziende.
Oltre al suo articolo, che risale all'ottobre 1999, esistono solo brevi saggi scientifci[2] che definiscono il problema del Predictive Dialing e, come soluzione, utilizzano sempre la stessa, il modello basato su "*ratio*".

2.2 Modello di Samuelson

Nel suo articolo, Samuelson afferma di essere l'invetore del Predictive Dialing. *[Rif: 8]*
Dopo anni di analisi di log di reali Call Center, e di numerose simulazioni, è riuscito ad individuare un algoritmo capace di incrementare il tempo di servizio di un agente fino a 57 minuti per ogni ora.
Nella sua dissertazione, Samuelson, spiega quali sono stati i vari stadi della ricerca ed arriva anche a descrivere, in maniera molto astratta, il modello che ha inventato.
Per ovvie ragioni di natura economica, la sua descrizione della soluzione proposta è parziale e sommaria. Non esistono né dettagli riguardanti il cuore dell'algoritmo, né altre utili informazioni per carpirne i segreti.

Un problema di tutti i sistemi di telemarketing sono sempre state le silent call.
Certi direttori dei Call Center, pretendevano un numero quasi nullo di silent call al giorno. Altri, più flessibili, ne ammettevano, al massimo, il 5% su tutte le chiamate delle ore lavorative. Samuelson era conscio di questo fatto, capì subito il principale trade-off intrinseco al predictive dialing.

---

[1] Douglas A. Samuelson è il Presidente della InfoLogix,Inc. Ha inventato il predictive dialing, è stato un imprenditore di successo ed ha collaborato anche con il Governo Americano come analista tecnologico. L'articolo è stato pubblicato sulla rivista Interfaces numero 29 del 1999. Il titolo è : Predictive Dialing for Outbound Telephone Call Centers.

[2] Tutti gli articoli a cui si fa riferimento, sono riportati in Bibliografia dal numero 7 al numero 11

Le possibilità erano due :

1. Garantire sempre un determinato valore minimo di minuti di lavoro per ogni ora, senza preoccuparsi delle silent call. Effettuando in maniera indiscriminata un elevato numero di telefonate verso i clienti.

2. Rispettare il limite imposto sulle silent call a discapito dei minuti lavorativi degli agenti. Effettuando tante telefonate quanti erano gli agenti disponibili.

Inizialmente, Samuelson monitorizza l'attività di grossi call center e, successivamente, individua delle misure da prendere in considerazione, per capire e valutare il comportamento degli stessi.
Tra gli indicatori scelti, i principali sono:

- fascia oraria di massimo e minimo traffico
- numero medio di squilli prima della risposta del cliente
- durata media delle telefonate
- tempo medio per effettuare una telefonata dalla composizione del numero al primo squillo

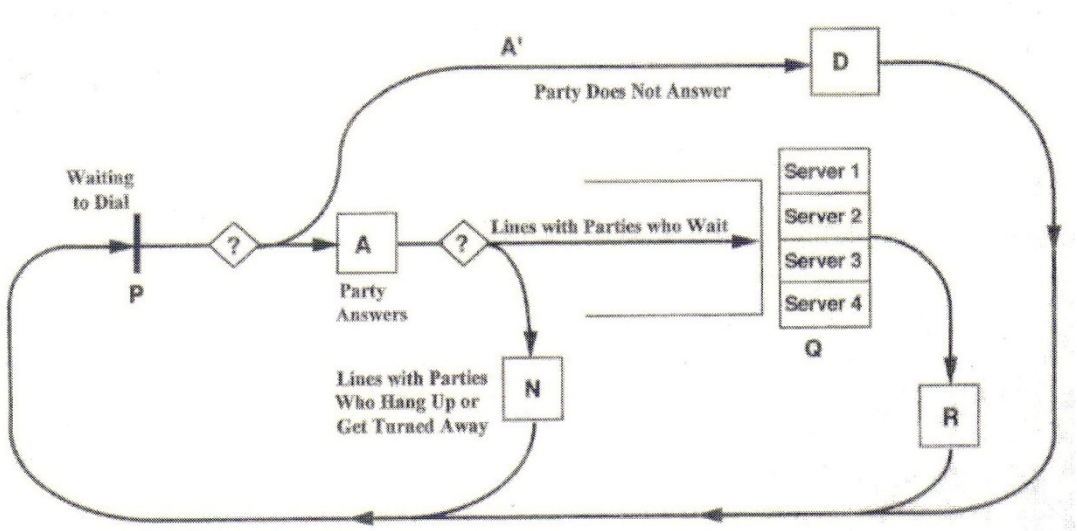
Inoltre, è stata importante anche l'analisi di tutti i log relativi ad errori interni al sistema stesso, oppure legati alla rete telefonica.
I suddetti indicatori, sono stati scelti in base a delle considerazioni non solo di ordine statistico ma anche pratico. Aveva scoperto che esistevano delle fasce orarie con maggiore probabilità di risposta da parte del cliente. Le chiamate non risposte, si concentravano soprattutto durante la mattina, mentre, tra le 5 e le 6 del pomeriggio, si quadruplicava la quantità di telefonate risposte dai clienti.
Una apparentemente ovvia ma importante considerazione, è stata quella di capire che molte campagne pubblicitarie venivano effettuate ad orari non adeguati. Ad esempio, era inutile chiamare la mattina, per una campagna che si rivolgeva esclusivamente al sesso maschile, poiché, durante il giorno, la maggior parte degli uomini era fuori casa per lavoro. Un fattore importante era anche la zona di destinazione della telefonata, la

distanza geografica influisce sui tempi di connessione della linea. Un Call Center presente a San Francisco, impiegava meno tempo per chiamare un cliente di Sacramento, piuttosto che uno a New York. Aveva capito che non c'era nessuna relazione tra la durata di una telefonata con quella della precedente, erano eventi indipendenti e separati. Anche i componenti hardware del Call Center giocavano un ruolo fondamentale. I tempi di composizione e di reset della linea erano differenti in base ai vari modelli in commercio in quegli anni.

Secondo quanto riscontrato dall'analisi dei log, Samuelson arriva alla conclusione che il sistema deve essere in grado di aggiornarsi in tempo reale. Ad esempio, in base alla fascia oraria, al tipo di campagna pubblicitaria e alla destinazione della telefonata, il modulo deve, in tempo reale, capire in che circostanza si trova e quindi regolarsi di conseguenza.

Dopo aver raccolto una grossa quantità di dati e dopo averli analizzati, Samuelson inizia lo studio del modello che sarebbe, poi, diventato il primo esempio di Predictive Dialing. Egli decide di strutturare il suo modello come un sistema chiuso, in cui sono presenti un certo numero di elementi in coda.

All'interno di questo sistema, ogni elemento della coda deve attraversare degli stadi ben precisi, come è visibile nella seguente figura.



(Figura 5) Le frecce rappresentano le linee telefoniche.

Al passo zero, il sistema effettua un certo numero di telefonate verso l'esterno su altrettante linee telefoniche. (P)
L'esito di queste telefonate può essere :

- Risposta (A)
- Non risposta (A')

Se *p* è la probabilità che si verifichi A, allora la probabilità che si verifichi A' è $q = 1-p$.

Se l'esito è *risposta*, allora si possono verificare ancora due casi :

- Telefonata terminata in aniticipo dal cliente oppure dal sistema, se non vi sono agenti disponibili (N)
- Telefonata servita da un agente (Q)

Se la telefonata viene interrotta, allora la linea telefonica viene liberata ed è pronta per una nuova telefonata (P), ma solo dopo un certo tempo.
Se la telefontata viene servita da un agente, allora la linea telefonica resterà occupata per tutta la durata della comunicazione (R) e, solo dopo il tempo necessario per il reset, torna ad essere pronta per una nuova telefonata (P).

Se l'esito è *non risposta*, allora la linea viene subito liberata e viene messa a disposizione per una nuova telefonata (P).

Durante tutta l'attività del Call Center, un sistema CTI registra tutti gli eventi che si verificano.
Tempi di inizio e fine telefonata, destinazione della stessa, tempi di composizione ed altri valori considerati utili per l'algoritmo. Sulla base dei dati registrati, in tempo reale, vengono calcolati medie, varianze, minimi, massimi e altri valori utili.
A questo punto interviene l'algoritmo innovativo di Samuelson che, partendo dai suddetti valori, calcola quante chiamate effettuare verso l'esterno in un dato istante.
In linea di massima, per ogni agente valuta quanto tempo è passato dall'inizio della chiamata. Confronta questo tempo con quello che il suo algoritmo ha predetto come durata massima. Se la differenza tra i due è minore o uguale al tempo necessario per

comporre un nuovo numero, allora incrementa di uno il numero di telefonate da effettuare.

Samuelson ha deciso di formulare la sua ricerca partendo da un modello, come quello mostrato in precedenza e non sfruttando un linguaggio matematico.
Era fermamente convinto che, partendo da una formulazione matematica del problema, avrebbe sicuramente dimenticato qualche importante fattore e/o vincolo, che, invece, è molto più evidente usando un modello siffatto.
Infatti, il suo modello è incentrato sui due maggiori vincoli del problema:

- Numero delle linee telefoniche disponibili
- Numero degli agenti disponibili

La coda Q degli agenti, è considerata nel modello, l'unico elemento a dimensione variabile, ma non illimitato. Le frecce che collegano i vari stadi del modello, rappresentano le linee telefoniche e sono considerate in numero fisso e non modificabile, durante l'attività del Call Center.

Tenendo in considerazione questi due vincoli e sfruttando le misurazioni introdotte in precedenza, Samuelson ha, quindi, prodotto un algoritmo che è in grado di individuare il numero ottimale di telefonate da effettuare verso l'esterno, per massimizzare il lavoro degli operatori, tenendo limitato il numero di silent call.

Dopo oltri due anni di prove e valutazioni, Samuelson individua dei valori numerici che migliorano ulteriormente il suo modello e che, nel 1993, gli hanno fruttato circa 12 milioni di dollari.
Ad esempio, dopo innumerevoli prove stabilisce che :

- il sistema deve aggiornare il suo stato interno (inteso come l'insieme dei valori numerici degli indicatori da lui stabiliti) ogni 10 minuti

- è utile effettuare 3 chiamate verso l'esterno per ogni agente, se la percentuale di telefonate che l'operatore porta a termine è inferiore al 20%

- è meglio usare, come criterio di valutazione, la durata massima di una telefonata piuttosto che la sua media

Samuelson ha anche provato ad assegnare, ad ogni chiamata, uno stato precedente alla vera risposta dell'operatore, detto *stato di acquisizione*, per cercare di aumentare il throughput del sistema. Questo esperimento ha dimostrato che non c'è nessun miglioramento, poiché, quasi sempre, risultava in una attesa inutile da parte del cliente che, quindi, terminava la telefontata con conseguente overhead dovuto al restart della linea stessa.

2.3 Modello basato sul fattore "*ratio*"

Il modello di Samuelson, è una versione raffinata di Predictive Dialer, che stabilisce il numero di chiamate da effettuare, sulla base di precisi ragionamenti, che seguono una logica comunque valida. Come detto anche in precedenza, la maggior parte dei Call Center, non potendo investire in ricerca, si sono adeguati tutti allo stesso modello standard. Per tutti, la soluzione più semplice da seguire era quella di sfruttare al massimo tutte le linee telefoniche, senza curarsi delle silent call.
Da una mancanza di fondi e da una intrinseca semplicità, è nato il *modello basato sul fattore ratio*. E' una soluzione assolutamente semplice che prevede di effettuare un numero di telefonate pari al numero di agenti, moltiplicato per un certo fattore, detto, appunto, *ratio*. Secondo questo modello, siano:

- t_s un qualsiasi valore dell'asse temporale
- j il numero massimo di linee telefoniche a disposizione
- Op il numero totale degli agenti
- q il tempo medio per comporre un numero telefonico
- R il valore della ratio

Il numero K<= *j*, di chiamate da effettuare all'istante $r_s = t_s - q$, per avere tutti gli O*p* agenti a lavoro è K = R* O*p*.

Generalmente R è un valore compreso tra 1.5 e 2 ed è scelto in modo da non poter mai ottenere un K maggiore di *j*. Con questa semplice ed immediata soluzione, è possibile sfruttare al massimo tutte le linee telefoniche ma, ovviamente, nessun risultato è garantito. E' un modello che non tiene minimamente conto dell'effetto silent call, che non utilizza nessuna predizione sulla possibile durata delle telefonate e che può portare ad un alto numero di chiamate inutili. Il suo vantaggio è sicuramente nel costo, che è praticamente nullo, e nella facilità di implementazione.

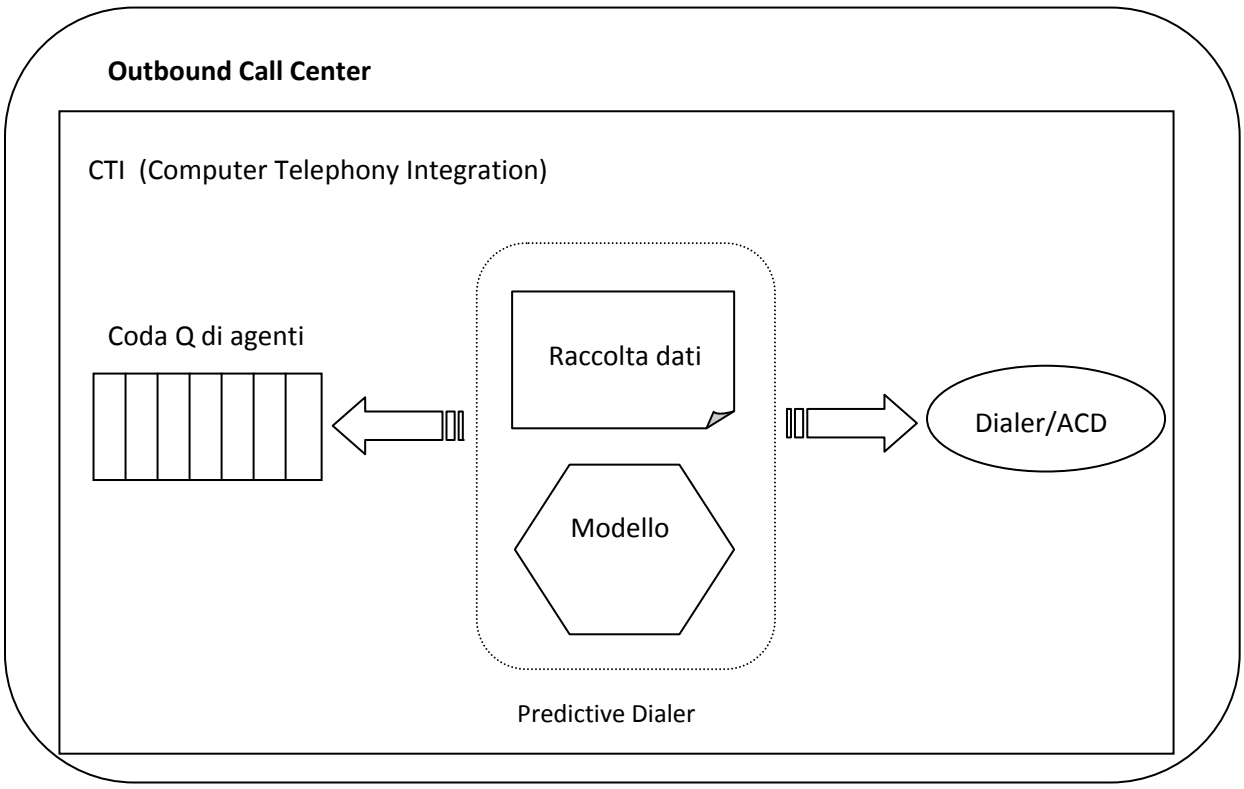
La dimostrazione del fatto che è sicuramente il metodo più diffuso, anche nei moderni Call Center, è che, in tutti i pochi articoli sugli outbound Call Center, viene sempre citato e ritenuto la versione standard di un predictive dialer di fascia bassa.

Capitolo 3

Modello proposto

3.1 Modello astratto

Sulla base della letteratura studiata sull'argomento, è stato sviluppato un modello per cercare di risolvere il problema del Predictive Dialing. Per descrivere il modello è stato utilizzato un approccio matematico/statistico, a differenza di quello più grafico di Samuelson.
In entrambi i casi, però, l'intero sistema Call Center può essere così schematizzato :



(Figura 6)

Questa schematizzazione dell'outbound Call Center è valida per tutti i sistemi di telemarketing che utilizzano il Predictive Dialing. L'unico punto di differenziazione è, sicuramente, la parte del modello, ovvero la logica utilizzata per ottimizzare il numero di telefonate da effettuare. Esiste sempre una coda Q di agenti, con un numero finito di elementi, ed un dialer hardware/software che effettua le telefonate verso l'esterno ed è in grado di smistarle agli agenti. Il collante tra queste due parti del sistema è il Predictive Dialer, che, interagendo con collegamenti CTI, ottiene i dati raccolti, li elabora, secondo il modello definito, e stabilisce quante chiamate effettuare in un particolare istante.

Nel caso specifico, il modello è stato pensato prima in forma astratta, per renderlo il più flessibile possibile e, successivamente, è stata introdotta una più dettagliata concretizzazione dello stesso.

Il modello astratto è scomposto in due sotto-modelli, strettamente dipendenti tra loro.

La prima parte del modello prevede l'esistenza di una funzione $y_1 = f_1(t_s)$, che ha come input un istante t_s qualsiasi e come output il presunto numero di agenti liberi in quel determinato momento t_s.
La funzione f_1 è quindi definita su tutto l'asse temporale ed ha come codominio l'insieme N dei numeri naturali. Per poter individuare il numero y_1 di agenti disponibili al tempo t_s, la funzione deve prevedere l'utilizzo di una distribuzione di probabilità, per stabilire la potenziale durata delle telefonate in corso. Sommando quest'ultima informazione al tempo di inizio della telefonata, è possibile stimare quando l'agente tornerà ad essere libero. Effettuando questo calcolo su ogni agente, è immediato stimare al tempo t_s, quanti e quali agenti potrebbero essere potenzialmente disponibili, oltre a quelli che lo sono sicuramente, perché non in conversazione.

La seconda parte del modello prevede l'esistenza di una funzione $y_2 = f_2(y_1)$, che ha come input il potenziale numero di agenti liberi nell'istante t_s. L'output della funzione è il numero di telefonate da effettuare verso l'esterno nell'istante $r_s = t_s - q$, con q tempo medio che il sistema impiega per effettuare il primo squillo verso un qualsiasi numero telefonico. Effettuando le telefonate nell'istante r_s, si garantisce, mediamente, che il

telefono squilli proprio nell'istante t_s di interesse. La funzione ha, quindi, sia come dominio che come codominio l'insieme N dei numeri naturali.
Nel caso ideale in cui :

- nel preciso istante t_s siano realmente disponibili y_1 agenti
- q sia l'esatto tempo che il sistema impiega sempre per effettuare il primo squillo della telefonata
- esistano sempre e solo telefonate con successo, ovvero risposte dal cliente

allora y_2 risulterebbe il numero ottimo ed unico, in base ai precedenti e determinati valori delle varibili in gioco, per soddisfare e risolvere il problema in questione.

L'algoritmo risolutivo, considerando il sistema già a regime, potrebbe essere il seguente:

```
int totalCalls,freeAgents , interval = 3000; //3 secondi
timer.interval = interval;

while(isOn){

     freeAgents= f1( Now + interval );

     if(freeAgents > 0){
          totalCalls = f2(freeAgents);
          timer.Start();
     }

     OnTimerElapsed{
          timer.Stop();
          ExecuteCalls(totalCalls);
     }

     OnEndCalls{
          isOn = false;
     }
}
```

Questo algoritmo è molto semplice e rappresenta un possibile *template* da utilizzare con qualsiasi implementazione delle due funzioni f_1 ed f_2.
E' stato volutamente pensato in maniera molto generale per essere utilizzato in contesti anche molto diversi. Ad esempio, non viene preso in considerazione come e dove vengono reperiti i numeri telefonici. Inoltre, non c'è alcun riferiemento al tempo medio impiegato per comporre un numero. Tutti i dettagli dipendenti strettamente dalle caratteristiche di uno specifico sistema sono stati omessi.

Il primo passo, è calcolare il potenziale numero di agenti liberi (*freeAgents),* passando alla funzione f_1, come parametro, l'istante per cui si vuole la predizione. Bisogna ricordare che il numero di agenti disponibili è il risultato della somma tra :

- agenti che sicuramente sono liberi, perché non hanno nessuna comunicazione in corso
- agenti che secondo l'algoritmo predittivo potrebbero liberarsi

La funzione f_2 viene eseguita solo se il numero di agenti disponibili risulta maggiore di zero. Il risultato sarà il numero di telefonate da effettuare verso l'esterno (*totalCalls)* nell'istante t_s. Un timer ha il compito di richiamare un metodo apposito per effettuare le telefonate. Solo nel caso ideale, il dialer inizierà la composizione dei numeri proprio nel momento t_s. In realtà, è consigliabile pensare più ad un intervallo, se pur minimo, in cui realmente il dialer inizia la composizione, piuttosto che al preciso istante t_s.

3.2 Modello proposto

Il modello astratto utilizza le funzioni f_1 ed f_2 senza specificare alcun dettaglio sulla loro implementazione. Questo ulteriore passo di raffinamento, si concretizza nel modello proposto, descritto di seguito.

A differenza degli anni ’90, oggi gli outbound Call Center sono in grado di gestire diverse campagne pubblicitarie contemporaneamente. Un agente può essere assegnato ad una o più campagne, spesso anche molto diverse tra loro. Ad esempio, una potrebbe essere un questionario ed un’altra una semplice promozione. I tempi medi tra le due sarebbero, generalmente, molto diversi.
Considerando questo fattore, la funzione f_1 ha bisogno di un ulteriore argomento per indicare la campagna pubblicitaria a cui si fa riferimento.
La funzione diventa, quindi, $y_1 = f_1(t_s, C_k)$ ed ha come output il possibile numero di agenti liberi nell’istante t_s, relativamente alla campagna pubblicitaria C_k.
Inoltre, in un caso reale e non ideale, gli esiti delle telefonate sono molteplici e possono essere suddivisi nelle seguenti categorie :

- occupato
- fax/fuori servizio
- senza risposta

Oltre a questi esiti autoesplicativi, ne esistono anche altri legati più al sistema call center che alla telefonata vera e propria, e possono essere raggruppati in una sola categoria :

- errori interni al sistema

Infatti, e’ possibile che un certo numero di telefonate, da effettuare verso l’esterno, non venga mai eseguito. Questo perché il sistema, per un qualsiasi motivo, può essere bloccato da uno o più errori hardware e/o software.

Detto ciò, anche la seconda funzione f_2 ha bisogno di un ulteriore argomento per indicare lo stato delle categorie sopra citate. In questo modo, è possibile ottenere un output coerente sia con il numero di agenti disponibili, che con l’andamento generale

del sistema. La funzione diventa, quindi, $y_2 = f_2(y_1 , S_{ts})$. Il suo output è il possibile numero ottimo di telefonate da effettuare, al fine di impegnare tutti gli y_1 agenti, considerando lo stato S_{ts} del sistema, al momento t_s.

Per la funzione f_1, sono state proposte due implementazioni, differenti solo per la distribuzione utilizzata. La prima si basa sull'utilizzo della distribuzione normale, mentre la seconda su quella normale inversa. Per entrambe le distribuzioni, sono state prese in considerazione le *funzioni densità di probabilità*. Il motivo dipende dal fatto di voler una stima della distribuzione su tutto un intervallo scelto, e non relativamente ad un valore massimo (funzione cumulativa) oppure prima o dopo un certo valore (funzione di ripartizione).

Di seguito l'implementazione con distribuzione normale.

In questa prima versione del modello, la funzione f_1 utilizza la distribuzione normale $N(\mu, \sigma^2)$, con media μ e varianza σ^2, per individuare in quale intervallo continuo, sull'asse temporale, si concentra maggiormente la durata delle telefonate. In questo caso l'evento (variabile casuale) è la durata della telefonata.

[Teorema del limite centrale : la somma di un numero di variabili indipendenti e distribuite casualmente è approssimabile ad una distribuzione normale se le variabili casuali hanno una varianza finita.][Rif : 12]

Sfruttando il Teorema del limite centrale, è possibile *ipotizzare* che la variabile casuale in questione sia caratterizzata da una distribuzione normale (o gaussiana), poiché :

- di sicuro le durate delle telefonate sono indipendenti tra loro
- il loro valore è continuo su tutto l'asse temporale e distribuito casualmente
- ogni evento si ripete più volte nelle stesse condizioni e quindi si *ipotizza* che possa avere una bassa varianza

Con queste premesse, è possibile utilizzare la funzione densità di probabilità, relativa ad una variabile casuale gaussiana, per individuare l'intervallo temporale di interesse :

$$p_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad \text{con} \quad -\infty < x < \infty$$

Non essendo possibile esprimere l'integrale della $p_X(x)$ in forma chiusa, mediante funzioni elementari, è necessario rendere disponibili in forma tabellare i valori della sua funzione di ripartizione. *[Rif :13, Normal Distribution, Characterizations with applications - Wlodzimierz Bryc]*

I più usati sono:

68,3% = P{ μ - σ < X < μ + σ }
95,0% = P{ μ - 1,96 σ < X < μ + 1,96 σ }
95,5% = P{ μ - 2σ < X < μ + 2 σ }
99,0% = P{ μ - 2,58 σ < X < μ + 2,58 σ }
99,7% = P{ μ – 3 σ < X < μ + 3 σ }

(Tabella 1)

Il modello prevede la suddivisione delle telefonate, relative alla campagna C_k, in classi di durata, espresse in secondi, su tutto l'asse temporale. Successivamente a questa suddivisione, avviene la scelta di un intervallo di durata $[t_1,t_2]$, per poter applicare la funzione densità di probabilità. L'intervallo viene scelto arbitrariamente, in modo da poter catturare il maggior numero di telefonate con una durata significativa.
Successivamente all'individuazione dell'intervallo, vanno calcolate la media μ e la relativa varianza σ^2 su tutte le durate comprese nello stesso.
Dopo aver calcolato i valori necessari, è possibile, tramite la *Tabella 1*, stabilire quale è l'intervallo temporale $[v_1,v_2]$ in cui si concludono la maggior parte delle telefonate, in relazione alla probabilità di successo scelta.
A questo punto, sapendo che un agente ha iniziato la sua telefonata nell'istante t_a, è possibile predire che, con una certa probabilità P, sarà nuovamente libero in un intervallo compreso tra $v_1 + t_a$ e $v_2 + t_a$.

Per avere una predizione sul numero totale di agenti disponibili nell'istante t_s, bisognerà sommare :

- numero di agenti sicuramenti liberi
- numero di agenti per cui, l'istante t_s, è compreso nell'intervallo di probabile teminazione $[v_1 + t_a, v_2 + t_a]$

Questa versione della funzione f_1, basata sulla distribuzione normale, ha un difetto che, su un grosso numero di telefonate, potrebbe essere influente in negativo.
In questo caso, per poter calcolare l'intervallo, a cui applicare la funzione densità di probabilità, bisogna scegliere un sottoinsieme di tutte le telefonate.
Infatti, viene scelto l'intervallo in cui terminano la maggior parte delle chiamate. Questo implica che, tutte le altre telefonate con durata non inclusa nell'intervallo scelto, vengono ignorate, come se non ci fossero. Questo tipo di sempliflicazione, porta ad una inevitabile imprecisione del modello e, quindi, a predizioni che posso essere completamente errate.

La seconda versione della funzione f_1, basata sulla distribuzione normale inversa, è stata introdotta proprio per evitare la perdita di precisione della versione precedente.
La distribuzione normale inversa è sembrata adatta al caso in questione perché, oltre ad essere continua, viene usata proprio per fenomeni che hanno la caratteristica di poter essere approssimati con la curva di Gauss con una coda molto più lunga.
In pratica, è usata nei casi in cui è presente una grossa quantità di valori approssimabile con la distribuzione normale, ma anche di numerosi altri che si dispongono oltre la campana di Gauss stessa. Tutto ciò sarà sicuramente chiarito con i grafici mostrati più avanti nel capitolo.
Nella seconda versione viene utilizzata la funzione densità di probabilità della normale inversa, che è la seguente :

$$\frac{\alpha\delta K_1\left(\alpha\sqrt{\delta^2+(x-\mu)^2}\right)}{\pi\sqrt{\delta^2+(x-\mu)^2}}\, e^{\delta\gamma+\beta(x-\mu)}$$

con :

μ = media all'interno della campana di Gaus (reale)

α = andamento della coda (reale)

β = parametro di asimmetria (reale)

δ = parametro di riduzione (reale)

[Rif : 14, The normal inverse Gaussian distribution: a versatile model forheavy-tailed stochastic processes, Hanssen, A., Oigard, T.A.]

In questo caso, la funzione densità di probabilità, non viene utilizzata come nel caso precedente. Questa volta viene utilizzato un approccio diverso. Tutte le volte che un agente inizia una telefonata, viene generato un valore *v,* a caso, che segue però la distribuzione normale inversa con ben determianti valori di μ, α, β e δ.
Il valore ottenuto viene considerato come una valida predizione per la durata della chiamata. Per aumentare ancora la probabilità di successo, in realtà, alla durata predetta viene aggiunto un margine *w* di errore. Quindi, la predizione completa riguarda, anche in questo caso, un intervallo di durata, che va da v-w a v+w.
Dopo aver calcolato questo intervallo, si procede esattamente come la versione precedente, per capire quanti agenti saranno liberi in un dato istante.

Per quanto riguarda la funzione $y_2 = f_2(y_1 , S_{ts})$, lo stato S_{ts} nel modello proposto è un insieme di valori percentuali. Questi ultimi indicano statisticamente quante sono, sul totale di tutte le chiamate effettuate, le telefonate con esito occupato, fax/fuori servizio oppure senza risposta e gli errori interni del sistema.
Da questo momento in poi, i suddetti valori percentuali saranno riferiti come *indicatori negativi,* poiché tendono a far diminuire il numero di telefonate con successo, ovvero risposte dal cliente.
Nel caso ideale, con y_1 agenti disponibili, basterebbe effettuare un numero di telefonate pari a y_1 per avere tutti gli agenti a lavoro. Sul totale delle telefonate effettuate, in un dato istante t_s, tutti gli esiti non positivi possibili sono espressi dagli indicatori percentuali di cui sopra. Se sono tutti a zero allora tutte le telefonate saranno risposte dal cliente e quindi risulteranno chiamate con successo.

Esiti telefonate = % successo + % numeri occupati + % fax/fuori servizio + % senza risposta + % errori interni

E' evidente che un aumento delle percentuali degli indicatori negativi porta ad un abbassamento delle telefonate utili, ovvero con successo.
Nei casi reali, per ovviare agli indicatori negativi, il modello prevede di sommare, al numero y_1 di agenti disponibili, i risultati degli indicatori negativi applicati al valore y_1 stesso. Tale risultato sarà il valore y_2, output della funzione f_2, nonché ipotetico numero ottimo di telefonate da effettuare nell'istante t_s.

3.3 Esempio di utilizzo del modello

I concetti espressi fino ad ora in questo capitolo, saranno chiariti con un esempio di utilizzo pratico del modello proposto.

Come si deduce da quanto scritto, sono necessari alcuni dati memorizzati dal sistema CTI, sull'intera attività del Call Center, per poter applicare il modello ed ottenere un comportamento affidabile.
Tra le informazioni necessarie, sono indispensabili :

- l'istante di inizio e di fine di ogni chiamata, per ogni agente, per ogni campagna
- il numero totale delle telefonate effettuate
- il numero totale delle telefonate con esito occupato, fax/fuori servizio e senza risposta
- il numero totale delle telefonate non riuscite a causa di errori interni al sistema

Avendo a disposizione i suddetti valori è possibile applicare il modello ad un caso reale.

Simboli :

- Op_j = operatore/agente generico J
- C_k = campagna pubblicitaria K
- μ_{ck} = durata media della telefonata relativa alla campagna pubblicitaria K
- σ^2_{ck} = varianza relativa alla campagna pubblicitaria K rispetto alla media μ_{ck}
- σ_{ck} = deviazione standard relativa alla campagna pubblicitaria K rispetto alla media μ_{ck} e alla varianza σ^2_{ck}
- B_{ck} = tempo di registrazione esito post telefonata relativo alla campagna pubblicitaria C_k
- Wop_j = valore booleano per indicare lo stato di pausa dell'operatore Op_j
- $N_{op\text{-}y}$ = stima del numero di operatori disponibili nell'istante Y

Dalla lista dei simboli si evincono due nuove voci, mai citate in precedenza, B_{ck} e Wop_j. Sono due valori che sono strettamenti legati al particolare call center e, quindi, non possono essere inseriti nel modello generale.

Il modello proposto è flessibile, non vincola nessun operatore ad una particolare campagna pubblicitaria ma, al contrario, è possibile assegnare più campagne pubblicitarie ad ogni agente.
Tutte le volte che un operatore Op_j inizia una telefonata con un cliente, relativamente alla campagna pubblicitaria C_k, viene memorizzato, dal sistema CTI, il preciso istante in cui si verifica tale vento. Lo stesso viene fatto al termine della stessa telefonata. Questa pratica si ripete per ogni operatore per ogni campagna. Al termine di ogni telefonata, l'operatore Op_j ha un tempo pari a B_{ck} per registrare nel sistema l'esito della chiamata, relativamente all'aspetto commerciale della stessa. Inoltre, ogni operatore Op_j ha la facoltà di potersi mettere in pausa per un certo tempo t_p e, quindi, di non essere considerato dal sistema come disponibile.

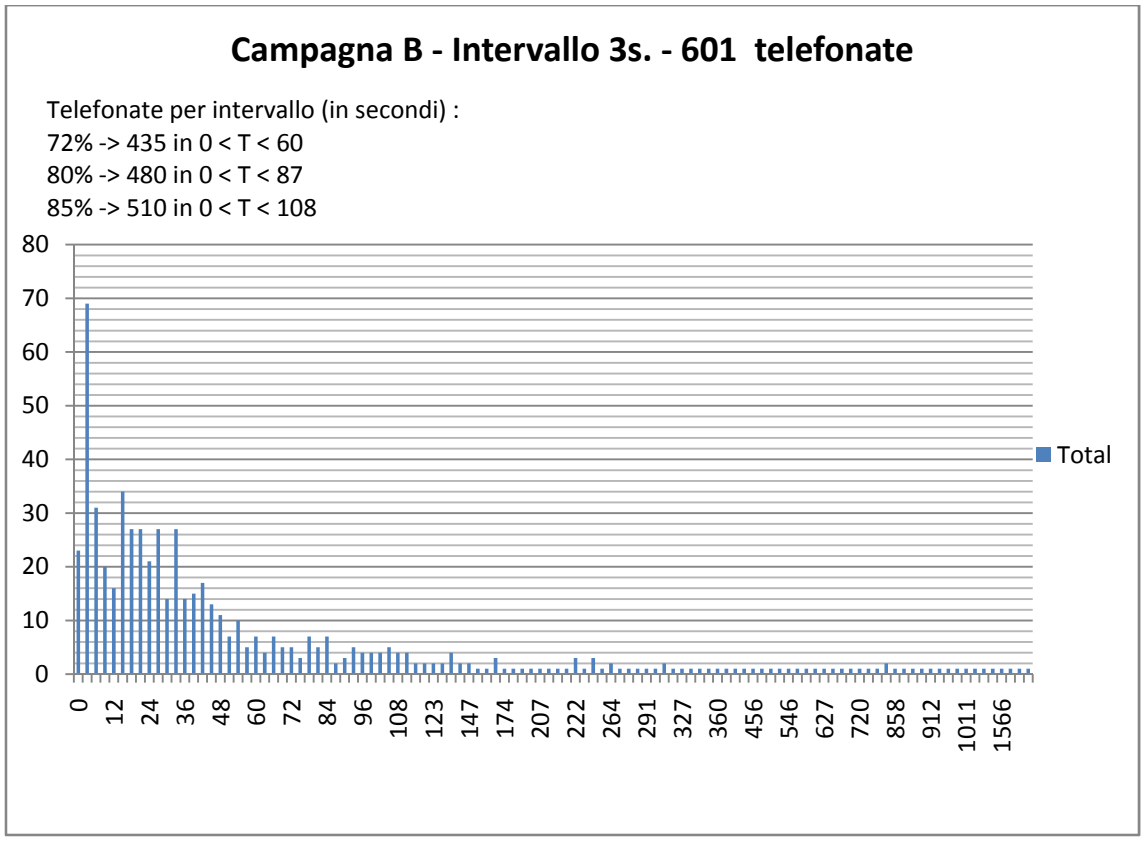
L'implementazione della funzione f_l prevede la suddivisione in classi di durata delle telefonate per ogni campagna. Intervalli da tre secondi ciascuno potrebbero essere una buona suddivisione per capire l'andamento delle durate delle telefonate.

I dati utilizzati di seguito, sono stati gentilmente forniti dalla sede di Firenze della società Lascaux[1].

La reale campagna pubblicitaria C_k presa in esame, è stata effettuata a partire dalle 9.00 del mattino fino alle 21.00 di sera.
Il numero totale di chiamate effettuato è stato 5.467, con i seguenti esiti (in seguito riferiti con R1) :

- 601 con successo (11%)
- 255 occupate (5%)
- 3299 non risposte (60%)
- 1244 fax (23%)
- 68 con errore (1%)

In questo caso, il valore B_{ck} è pari a 15 secondi.
Di seguito il grafico che riporta sull'asse X gli intervalli temporali, di durata da tre secondi ciascuno, e sull'asse Y il numero di telefonate che è terminato in ogni intervallo.
(Grafico 1)



[1] Lascaux nasce nel 2004 ad Arezzo nel campo dell'lnformation Technology.Rappresenta oggi una realtà Toscana con più di 30 collaboratori, 2 sedi operative sul territorio regionale, progetti avviati sul territorio nazionale ed europeo

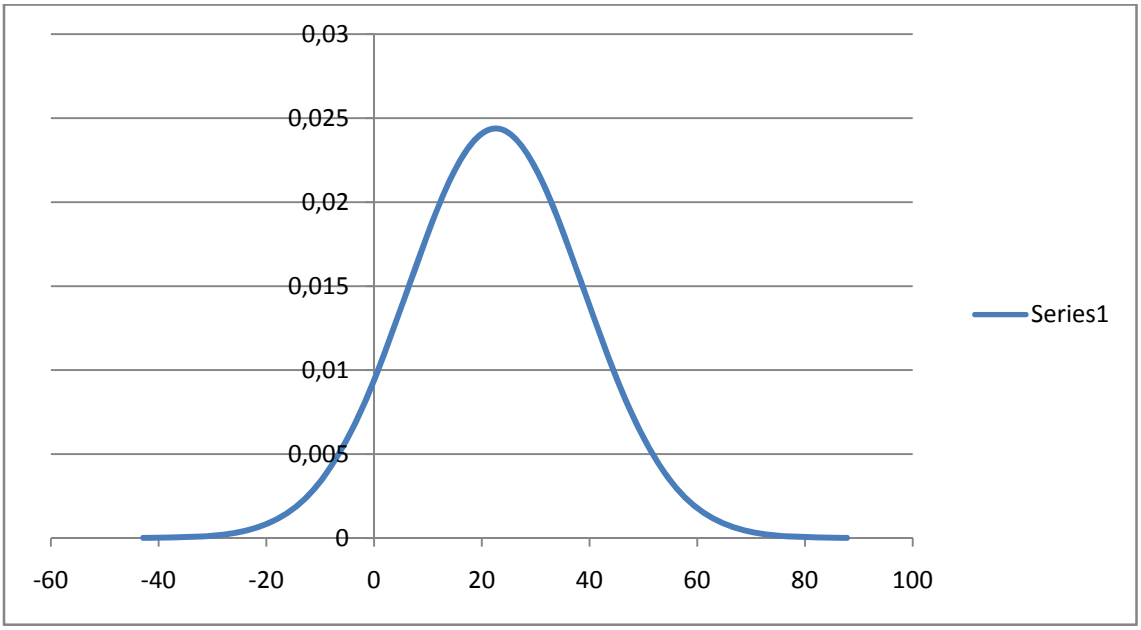
Soluzione valida con f_l implementata con distribuzione normale.

Dal grafico è possibile affermare che la scelta dell'utilizzo della distribuzione normale potrebbe essere adeguata, se si prende in considerazione l'intervallo [0,60], in cui ricadono il 72% delle telefonate totali.
La durata media (μ) all'interno dell'intervallo scelto è di circa 23 secondi. La varianza (σ^2) è circa 268.

La soluzione tabellare della funzione di ripartizione, nel caso in questione, è la seguente:

68,30%	6,222418	< X <	38,9499963
95,30%	-9,48682	< X <	54,6592341
95,50%	-10,1414	< X <	55,3137857
99%	-19,6324	< X <	64,8047835
99,70%	-26,5052	< X <	71,677575

Con la seguente rappresentazione grafica :



(Grafico 2)

A questo punto è possibile predire che, ad esempio, con il 95,5% di successo, la durata delle successive telefonate sarà compresa tra 0 e 55 secondi.

Con questa informazione, si può ipotizzare che un agente che inzia una telefonata alle 11.00, ad esempio, potrà essere libero nell'intervallo [11.00,11.55 + 15], dove 15 secondi rappresentano il tempo B_{ck} utile per registrare l'esito commerciale della chiamata.

Poiché, di volta in volta, per ogni agente, viene eseguito questo calcolo, è facile sapere in un certo tempo t_s , quali e quanti potrebbero essere i potenziali agenti a disposizione, a meno del valore Wop_j, relativo allo stato di pausa dell'agente.

Dopo aver calcolato il valore y_1 è possibile applicare la funzione f_2 per individuare il numero ottimale di chiamate da effettuare.

Se tutti gli indicatori negativi di R1(vedi sopra), fossero pari allo 0% e fossero tutte telefonate con successo (caso ideale), allora il numero ottimale di chiamate y_2 da effettuare, sarebbe proprio y_1.

Essendo in un caso reale, il modello prevede di sommare ad y_1 i valori percentuali degli indicatori negativi ad y_1 stesso.

Nel caso in questione si ha :

$y_2 = y_1 + (5\% + 60\% + 23\% + 1\%)$ di $y_1 = y_1 + 89\%$ di y_1

Il risultato finale dell'applicazione del modello è, quindi, y_2. Ovvero, il numero di telefonate ottimo da effettuare nell'istante t_s per avere tutti gli agenti a lavoro.

Soluzione valida con f_1 implementata con distribuzione normale inversa.

Per cercare di catturare tutte le chiamate, non solo quelle comprese nell'intervallo [0,60], bisogna utilizzare la distribuzione normale inversa.

I valori dei parametri utilizzati per la funzione densità di probabilità sono i seguenti :

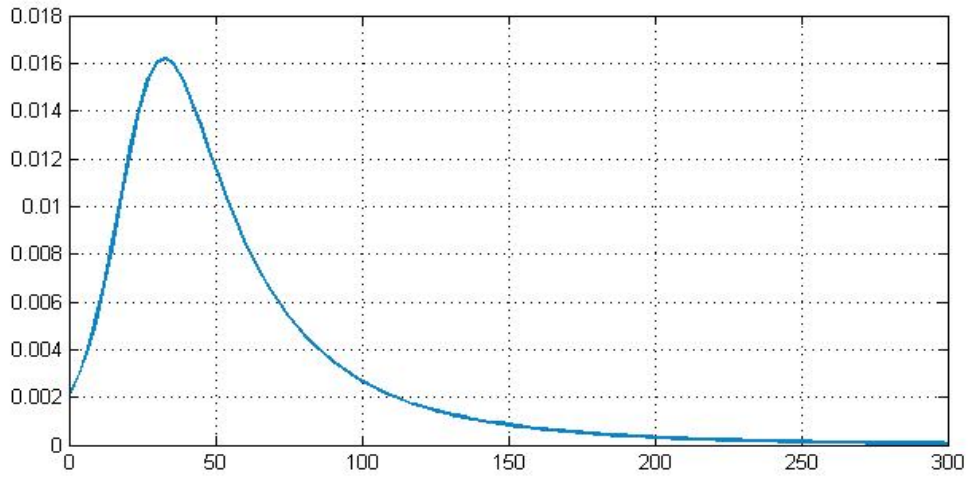
$\mu = \mu_{ck} = 23$ s.

$\alpha = 0.05$

$\beta = 0.04$

$\delta = 25$

Il grafico ottenuto per la campagna presa in esame è il seguente :



(Grafico 3)

Per motivi di leggibilità è stato riportato solo l'intervallo [0,300]. Per tutti i valori che seguono, fino all'ultimo, l'andamento approssima con buona precisione quello del grafico delle durate (Grafico 1).

Questa è la dimostrazione che la distribuzione normale inversa può essere utilizzata per approssimare il reale andamento della distribuzione delle durate. A differenza della distribuzione normale, in questo caso non venogono effettuate eccessive approssimazioni.

Per i suddetti motivi, il modello, in questo caso, ha un comportamento diverso rispetto al precedente. Se un agente inizia la telefonata alle ore 11, viene fatta una predizione sulla base di una produzione casuale di un valore *v,* che rispetta la funzione densità di probabilità stabilita. Prendiamo in considerazione un intervallo di errore di +/- 2 secondi. Quindi, l'agente sarà libero nell'intervallo $[11+v - 2, 11+v + 2]$. Se il tempo t_s è contenuto nell'intervallo individuato, allora l'agente è da ritenersi libero.

Questo calcolo viene effettuatato per ogni agente, per ogni campagna, proprio come nel caso precedente. La somma di tutti gli agenti sicuramente liberi e di quelli liberi secondo la predizione, sarà poi utilizzato dalla funzione f_2 che mantiene la stessa logica vista in precedenza.

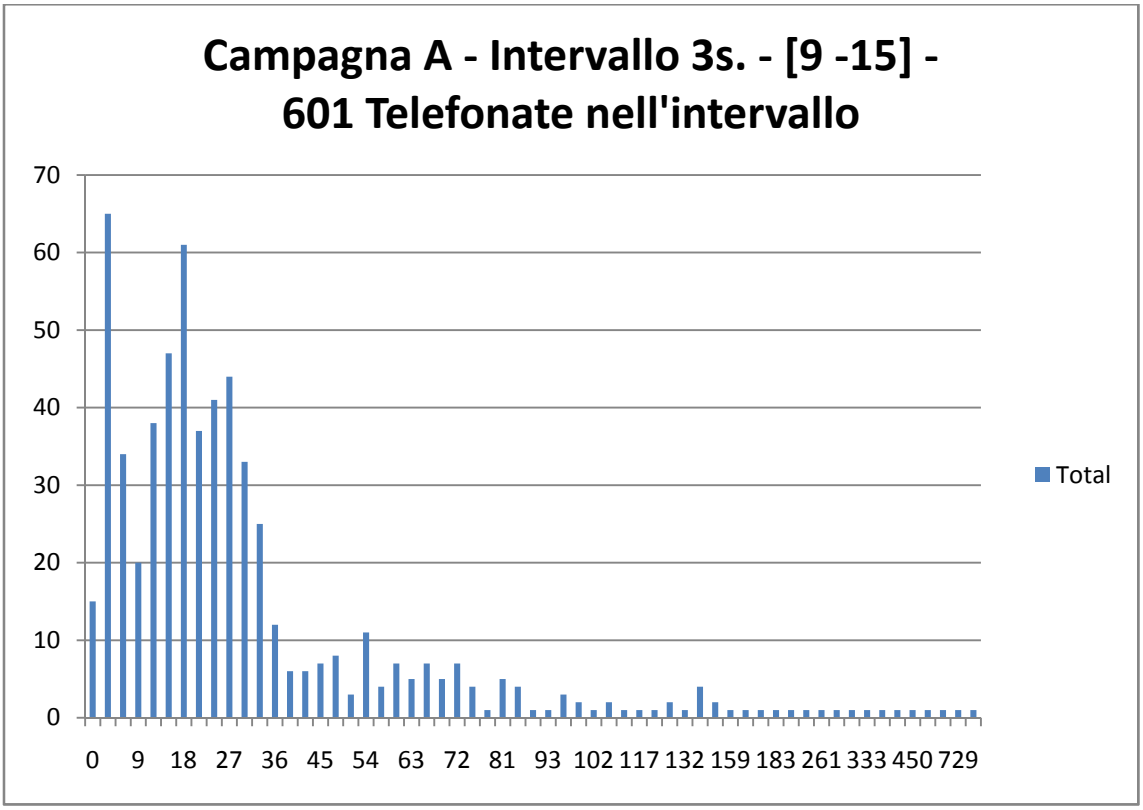
E' importante una puntualizzazione su come stabilire il valore μ da utilizzare per la funzione densità di probabilità. Questo valore indica la "localizzazione" della distribuzione, ovvero quale è il valore intorno a cui si concentrano la maggior parte dei dati. Poiché questo tipo di distribuzione, come detto in precedenza, è un tipo particolare di distribuzione normale, il valore μ è la media dei valori all'interno dell'intervallo che è sottostante la campana di Gauss.

3.4 Ottimizzazioni del modello

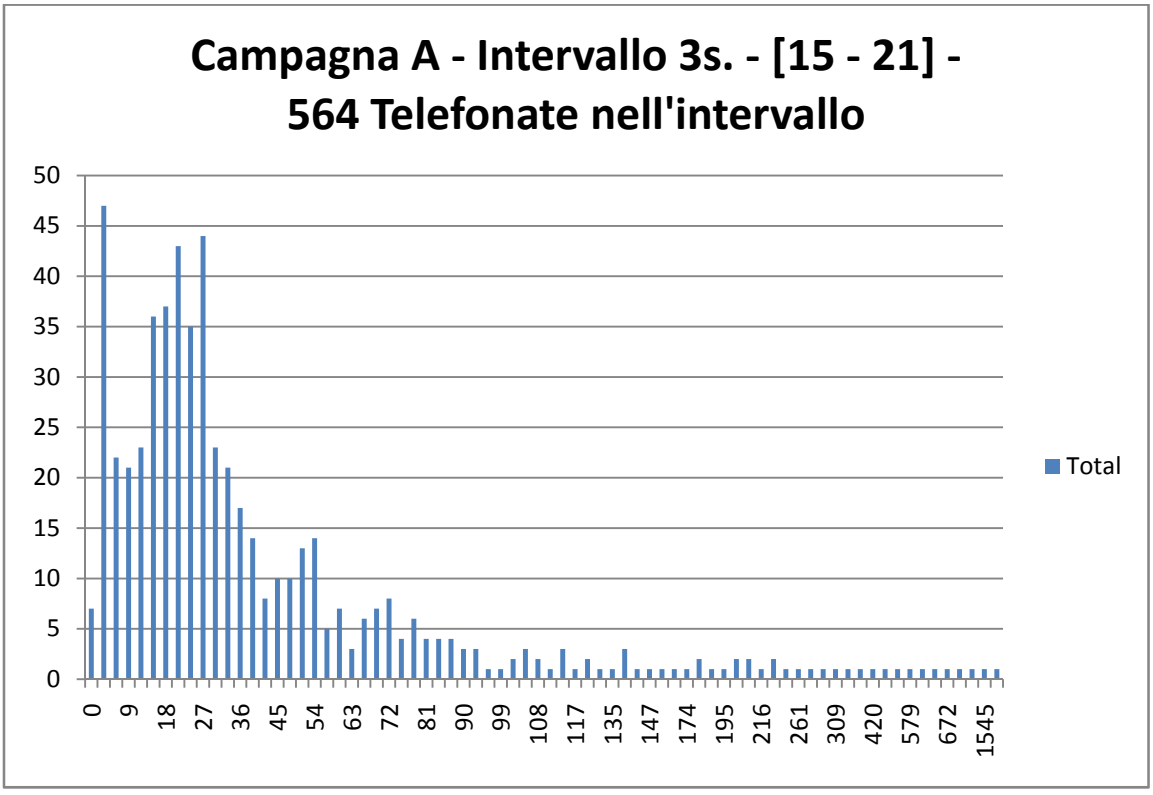
Per affinare il modello, si potrebbero introdurre nuovi indicatori per stabilire con maggiore accuratezza il numero ottimo di chiamate da effettuare in un dato istante.
Il principio è che tante più informazioni vengono utilizzate nella funzione f_2 e tanto più preciso dovrebbe risultare il modello.

Sicuramente sarebbe plausibile pensare che la fascia oraria possa influire sulla durata media delle telefonate. E' ipotizzabile che durante particolari fasce orarie, come le ore dei pasti ed inizio/fine giornata, ci possa essere una maggiore propensione da parte del cliente a terminare subito la telefonata.
In realtà, questa ipotesi è stata smentita analizzando i dati relativi a varie campagne pubblicitarie reali, per un totale di circa 522.000 telefonate.
Tutte le chiamate sono state suddivise in classi di durata da tre secondi ciascuna. Successivamente è stata valutata la distribuzione della durata media su due fasce orarie differenti : 9-15 e 15-21. Questo tipo di analisi, ha dimostrato che la distribuzione normale inversa della durata delle telefonate, persiste anche sulle fasce orarie e che non c'è relazione tra l'orario in cui viene effettuata una telefonata e la sua durata.
Sulla base dei risultati ottenuti, non c'è modo per ottimizzare il modello, sfruttando le fasce orarie e mettendole in relazione con le durate delle telefonate.

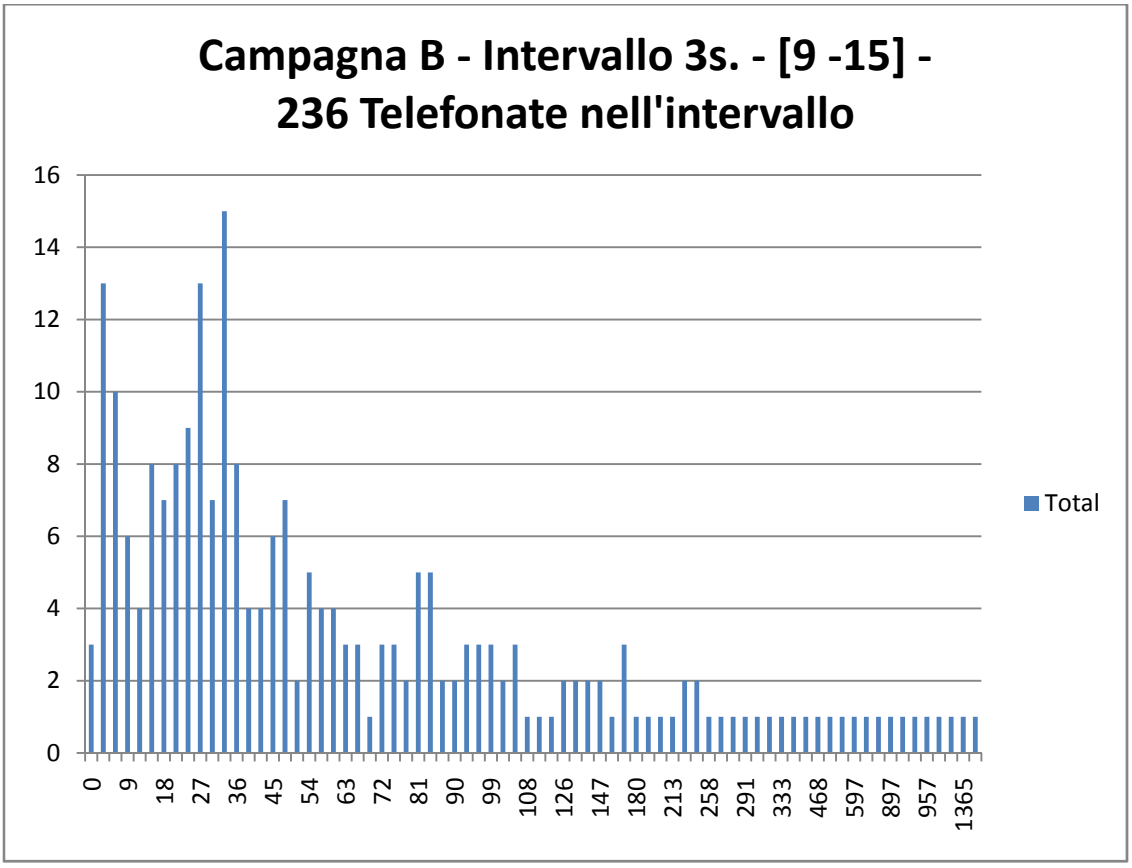
I seguenti grafici mostrano i risultati dell'analisi della durata delle telefonate rispetto alla fascia oraria. Sull'asse X gli intervalli temporali da tre secondi ciascuno e sull'asse Y il numero di telefonate che è terminato in ogni intervallo.



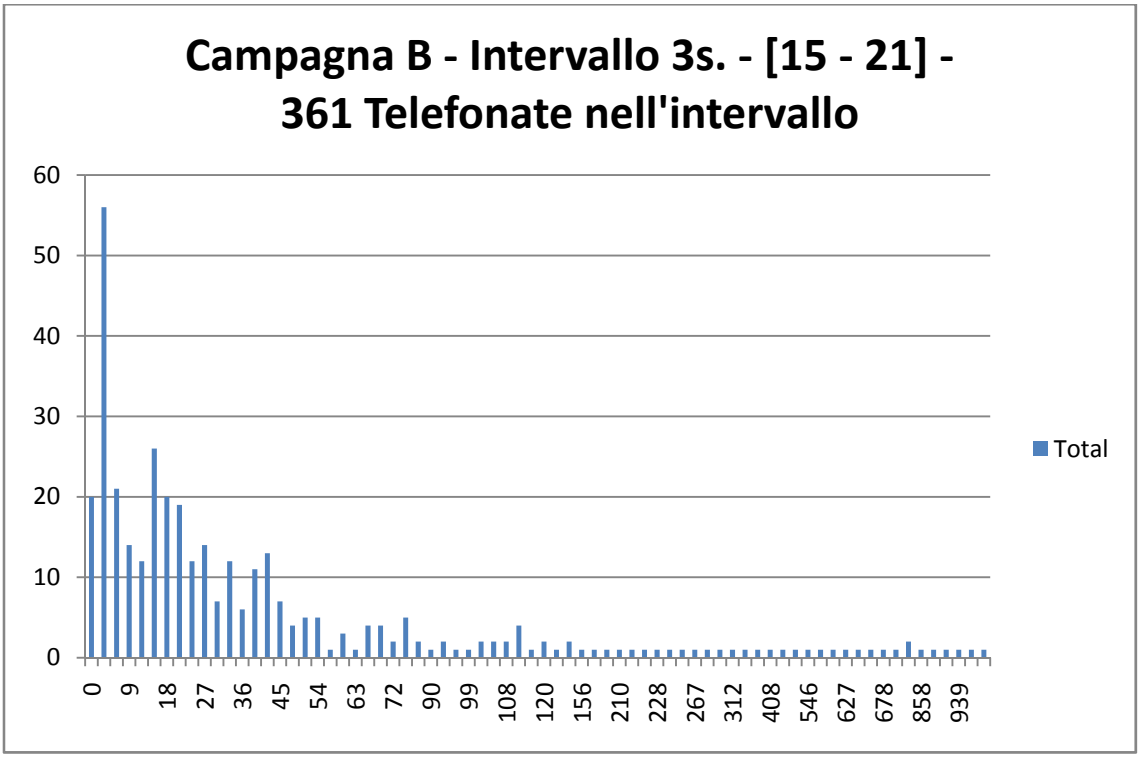
(Grafico 4)



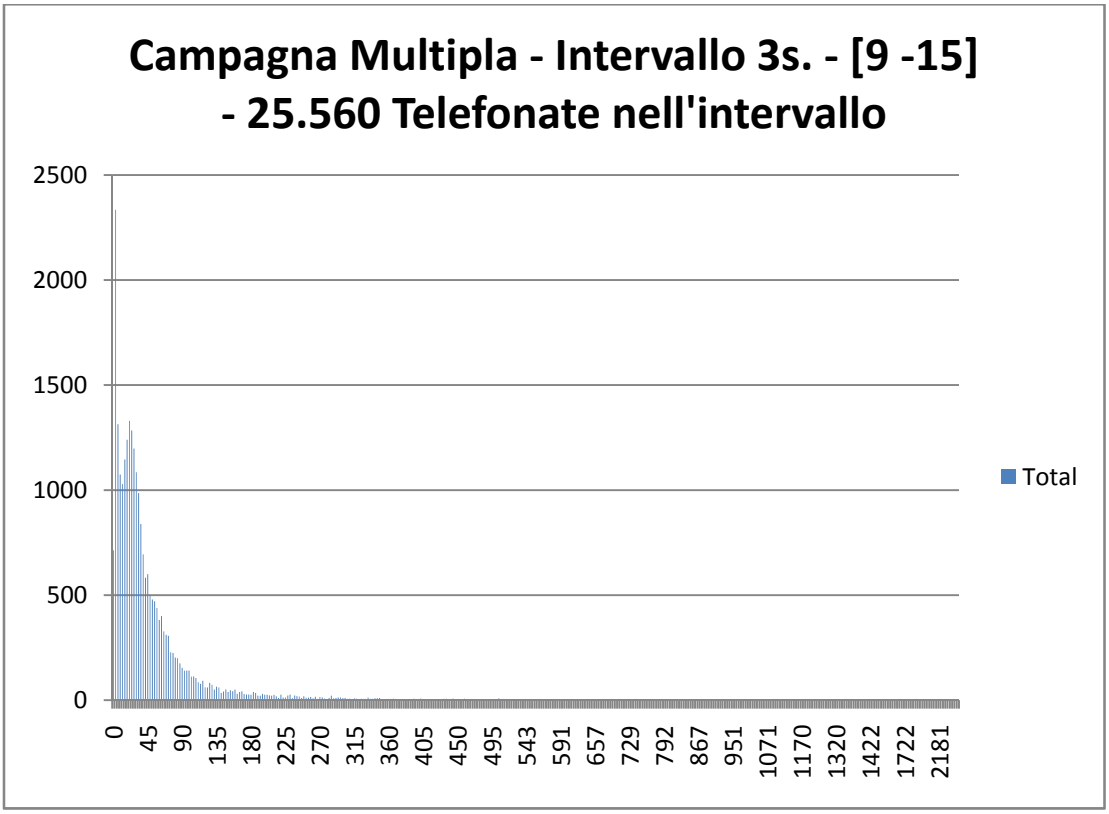
(Grafico 5)



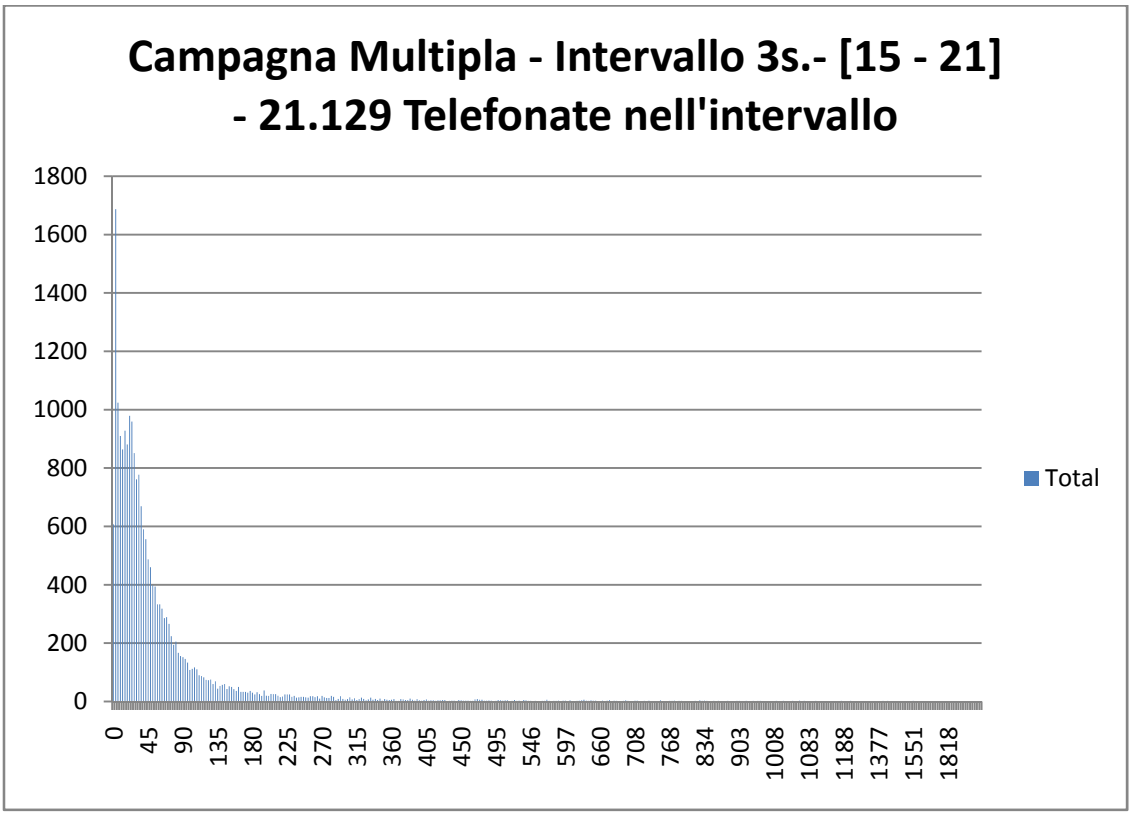
(Grafico 6)



(Grafico 7)



(Grafico 8)



(Grafico 9)

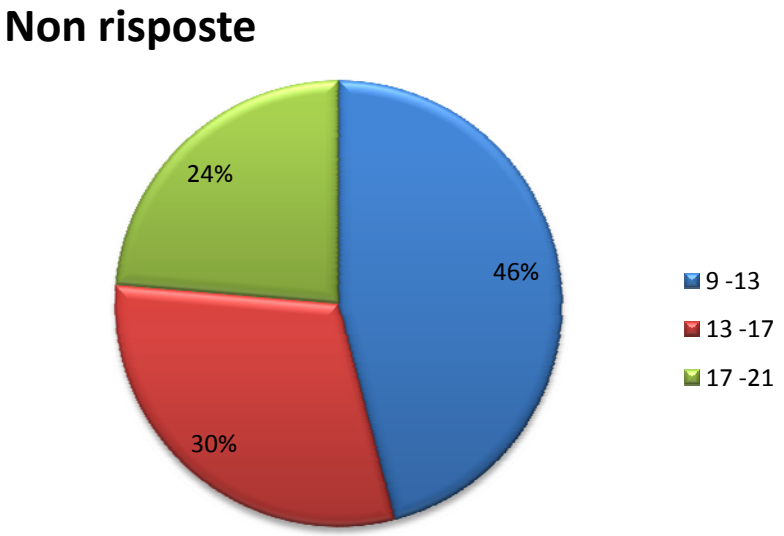
Dai grafici è possibile subito notare che, indipendemente dalla fascia oraria, la durata delle telefonate ha sempre la stessa distribuzione. La maggior parte di esse, termina tra nell'intervallo compreso tra 0 e 100 secondi.

Potrebbe essere utile cercare una relazione tra la fascia oraria e gli indicatori negativi. Tra questi ultimi, potrebbero essere legati all'orario, il numero di telefonate occupate ed il numero di chiamate non risposte da parte dei clienti. Gli altri indicatori (fax/fuori servizio,errori interni) si presume che non abbiano alcun legame con l'orario, per cui non verranno presi in considerazione per questa ulteriore analisi.

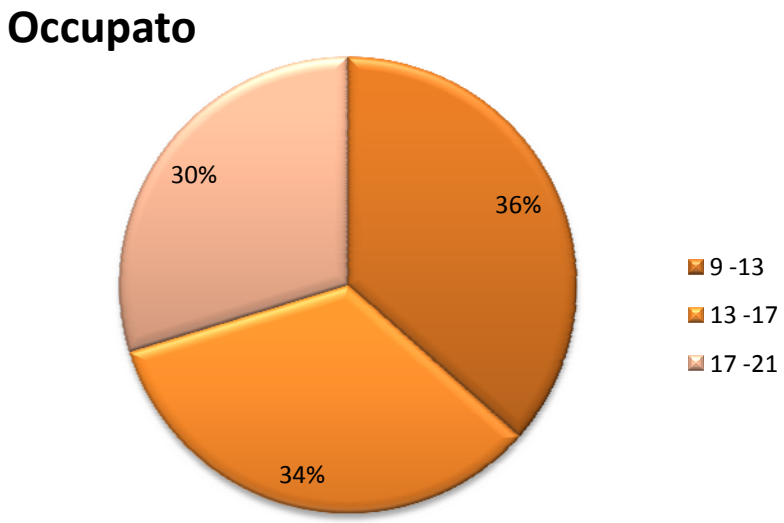
Su un totale di 522.000 telefonate, 197.990 sono risultate non risposte dai clienti (pari a circa il 38%) e 23.624 sono risultate occupate (pari al 4.5%).
Per capire se la fascia oraria influisca o meno su questi due indicatori negativi, le telefonate occupate e non risposte sono state suddivise in base a tre fasce orarie : 9 – 13, 13 – 17 e 17 - 21.

I risultati hanno dimostrato che sia le telefonate non risposte che quelle risultate occupate hanno una tendenza ad essere maggiormente raggruppate nella fascia oraria 9 - 13. Questa osservazione è valida soprattutto per le chiamate non risposte.
La probabile spiegazione logica di questo fenomeno, è legata al fatto che durante la mattina le persone sono a lavoro e non in casa.
Infatti, mettendo insieme i risultati di tutte le quasi 198.000 chiamate non risposte si ha che il 46% di esse si concentra tra le 9 e le 13. Mentre, nella stessa fascia oraria, si concentra il 36% delle chiamate occupate.

Osservando i grafici sottostanti, si nota, come nel caso delle chiamate occupate, che la fascia oraria non sia molto importante, ed è quindi trascurabile come parametro di ottimizzazione.



(Grafico 10)



(Grafico 11)

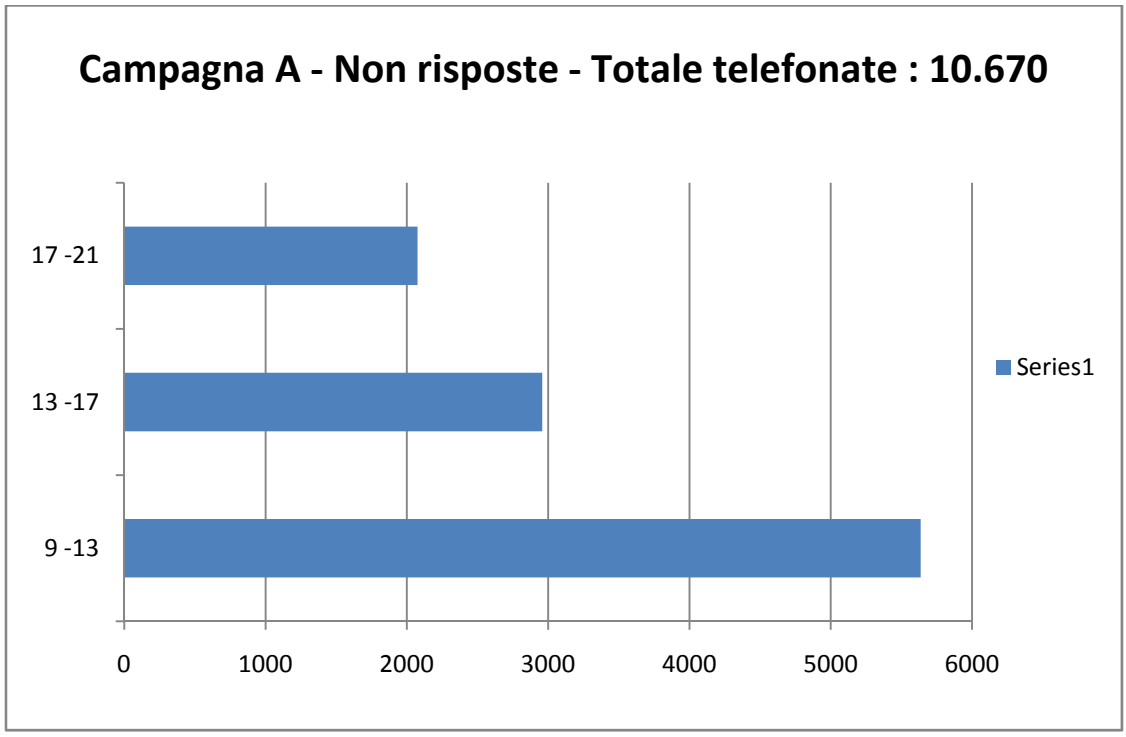
Per quanto riguarda i numeri non risposti, invece, è il contrario. La differenza percentuale tra la prima fascia e le altre due è importante e, quindi, l'orario potrebbe essere una informazione utile per ottimizzare il modello relativamente all'indicatore negativo "non risposte".

Per sfruttare questa informazione, il modello potrebbe essere modificato relativamente alla funzione f_2.

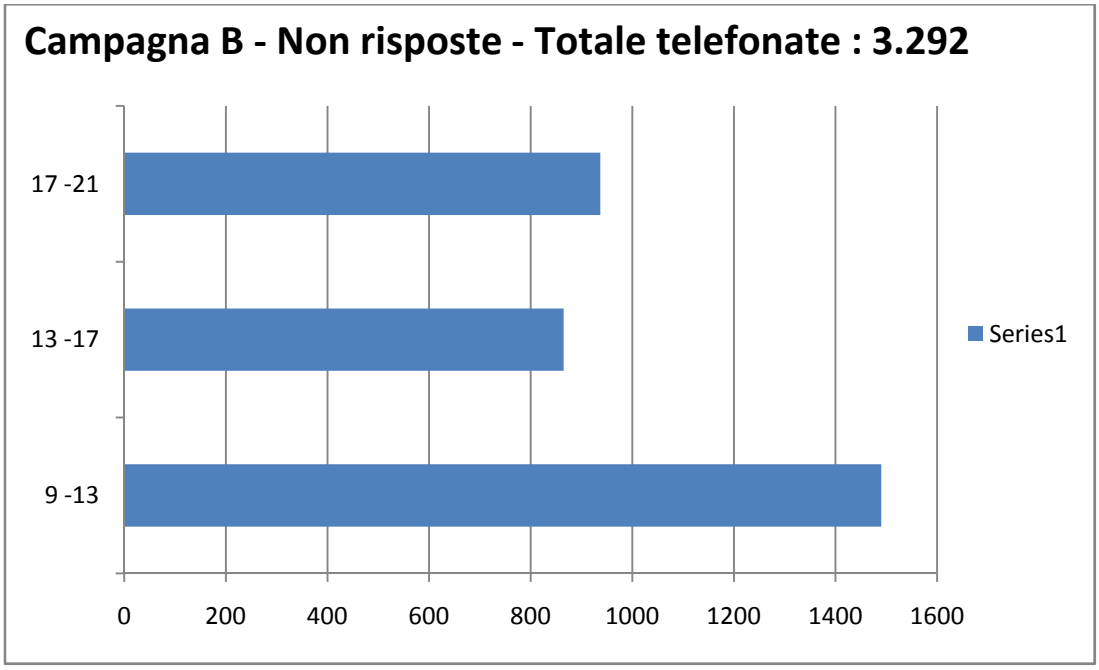
Durante il calcolo, per l'indicatore "non risposte" sarebbe meglio utilizzare la percentuale relativa alla fascia oraria in cui ricade il momento t_s ,e non il valore medio distributi su tutte le 12 ore di lavoro del call center.

Di conseguenza, lo stato S_{ts} , non dovrà contenere un solo varole percentuale per le chiamate non risposte ma, in realtà, i valori saranno tre, un valore per ogni fascia oraria. Di volta in volta, la f_2 utilizzarà il valore percentuale adatto in base all'orario.

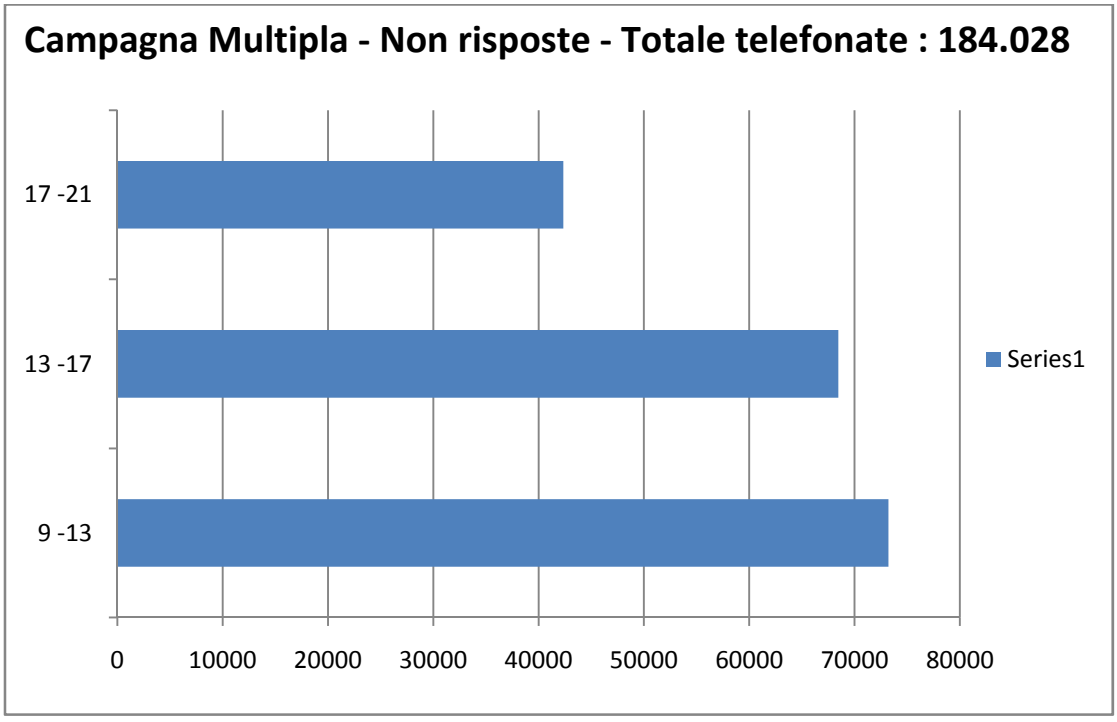
I seguenti grafici mostrano i risultati dell'analisi della durata delle telefonate rispetto alla fascia oraria. Sull'asse X il numero di telefonate e sull'asse Y le fasce orarie di interesse.



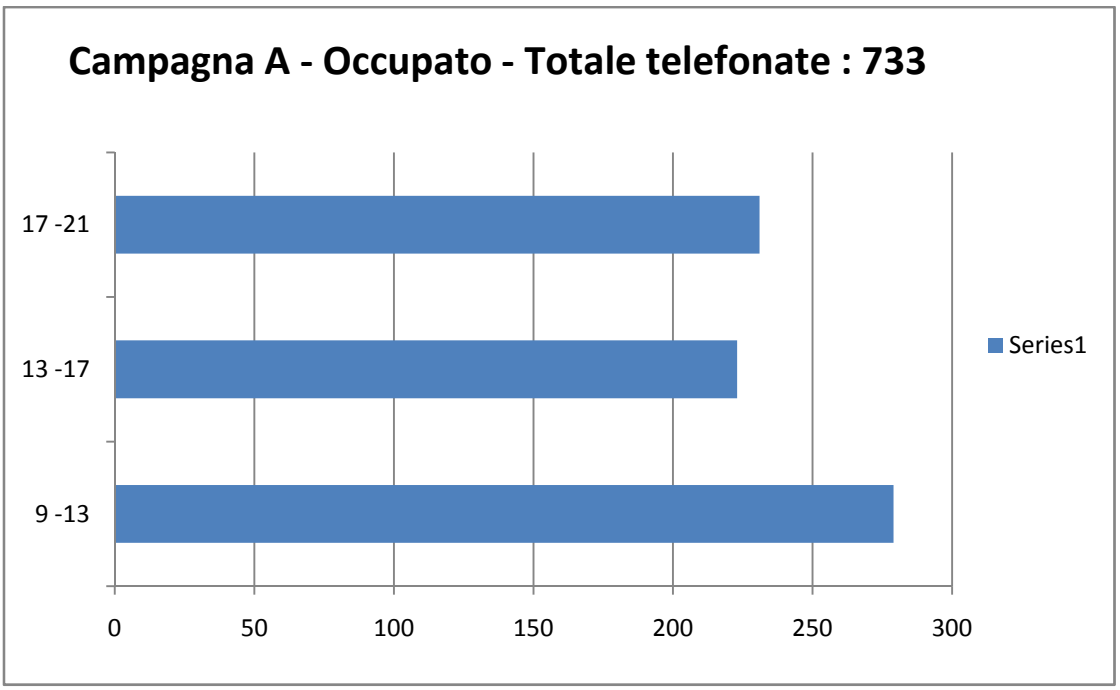
(Grafico 12)



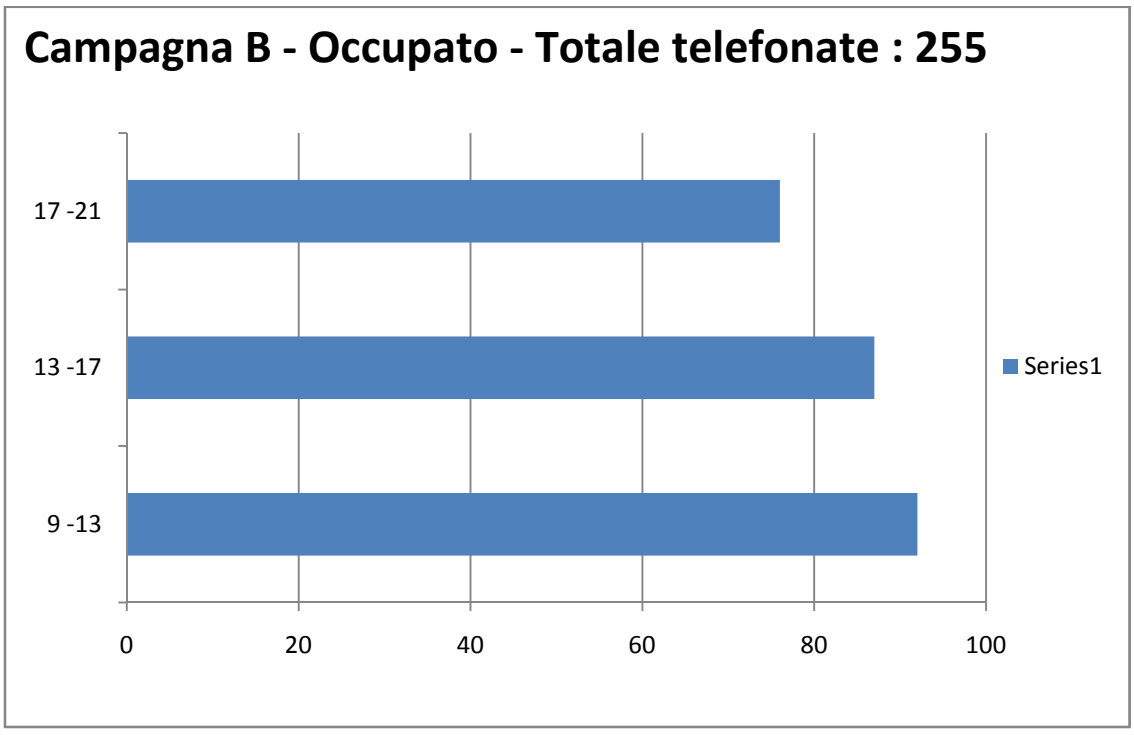
(Grafico 13)



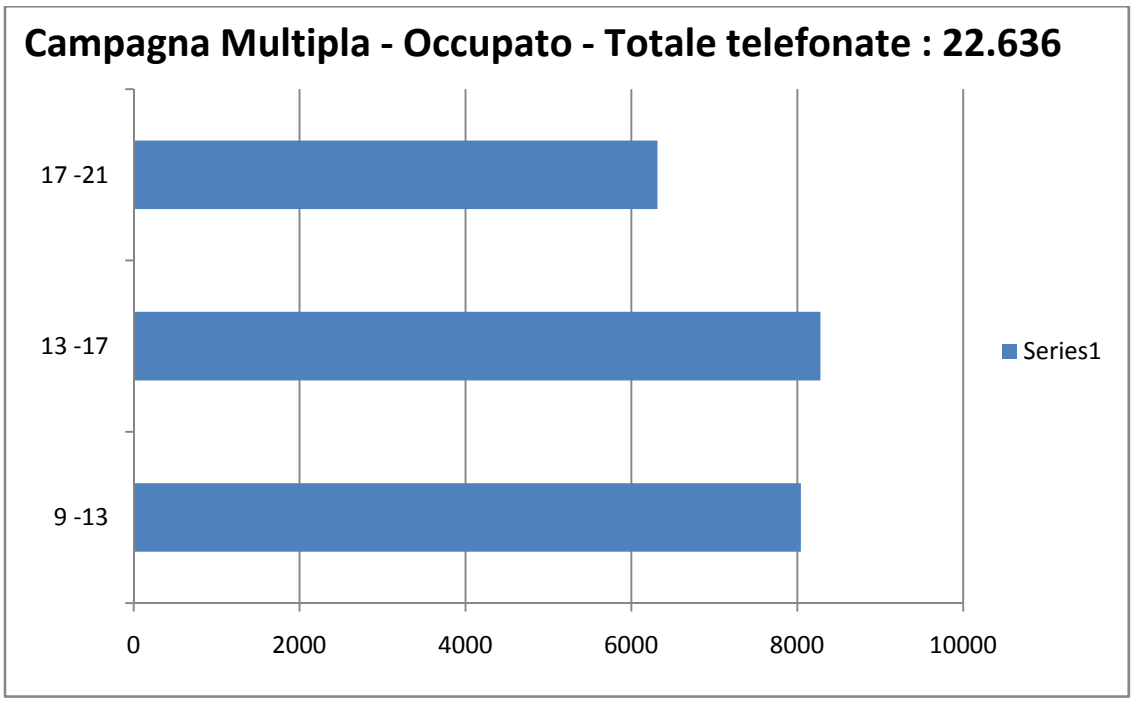
(Grafico 14)



(Grafico 15)



(Grafico 16)



(Grafico 18)

3.5 Confronto teorico con i modelli esistenti

Dopo aver introdotto i due principali modelli presenti in letteratura sull'argomento, ed aver proposto un modello alternativo, è necessario un confronto tra gli stessi.

Il modello più semplice è senza ombra di dubbio quello basato sul fattore "ratio".
E' un modello che può essere utilizzato senza eccessivi costi di gestione e manutenzione. Non prevede alcun tipo di calcolo o analisi complicata prima di poter essere messo in funzione. Proprio per questi motivi, e' sicuramente il sistema più diffuso. L'unico calcolo da effettuare riguarda il numero di linee a disposizione per l'attività di telemarketing. In relazione alla quantità di linee di cui si dispone, bisogna regolare il fattore "ratio" utilizzato, per non cercare di comporre più numeri di quanti in realtà siano possibili. Non utilizzando alcun particolare algoritmo predittivo, la sua riuscita è completamente affidata al caso. In teoria potrebbe essere il migliore algoritmo

in circolazione se, effettivamente, tutte le chiamate effettuate risultassero risposte dai clienti. In questo caso si avrebbe un numero pari a zero di silent call ed il 100% di produttività per gli agenti. Allo stesso modo, però, potrebbe risultare un pessimo algoritmo. Se tutte le telefonate fatte in più, rispetto al numero di agenti disponibili, fossero telefonate comunque risposte dai clienti, si avrebbe un elevatissimo numero di silent call, senza però aumentare la produttività degli agenti.
In realtà non è possibile fare un vero e proprio confronto tra questo modello, quello di Samuelson e quello proposto. Non è possibile fare alcun tipo di valutazione sui differenti algotirmi di predizione perché, il modello con ratio, non ne ha uno vero e proprio. Il confronto, però, può essere sicuramente valido tra i restanti due modelli.

Prima di tutto, è corretto dire che nell'articolo di Samuelson non viene descritto il modello in dettaglio. Per cui, il confronto è valido solo limitatamente a quello che stato scritto e a quello che si può dedurre.
La prima differenza riguarda il diverso approccio utilizzato. In un caso, Samuelson imposta il problema come lo studio di un modello grafico con precisi vincoli, mentre, nel caso del modello proposto, il problema è affrontato più da un punto di vista matematico e statistico.

Esistono dei punti comuni e di contatto tra i due modelli.
In entrambi i casi, il modello prevede l'utilizzo e l'analisi dei dati raccolti dal sistema CTI. Alcuni tra gli indicatori presi in esame sono gli stessi, come :

- fascia oraria
- durata delle telefonate
- tempo medio per effettuare una telefonata dalla composizione del numero al primo squillo

Quindi, per tutti e due i modelli, è fondamentale avere una grossa quantità di dati da cui imparare per poter essere attendibili.

Nel modello di Samuelson il problema viene trattato come unico blocco, in cui i vincoli principali sono il numero delle linee telefoniche ed il numero degli operatori.
I vari stadi del sistema sono, in realtà, macro-stadi che rendono il modello poco flessibile. Ad esempio, nello stadio D, le linee non risposte vengono subito liberate per effettuare nuove chiamate, ma non c'è nessuna distizione sul motivo della non risposta. Se il numero fosse occupato sarebbe più sensato non liberare la linea e ritentare subito la telefonata.
Negli anni in cui Samuelson ha introdotto il suo modello, non era consuetudine l'esecuzione di diverse campagne pubblicitarie nello stesso momento.
Questo stato delle cose si riflette nel modello, che non fa nessuna distinzione sul tipo di campagna effettuata dall'operatore. Tutti gli operatori sono visti come entità uguali, quando, in realtà, questo è un grande errore. I tempi di un agente che effettua un questionario, sono diversi dai tempi di un suo collega che spiega una promozione per un prodotto. Non esiste né la possibilità di valutare il modello in un contesto in cui vengono effettuate più campgna pubblicitarie, né l'eventualità di dividere gli agenti in base all'esperienza e alle capacità.
Così facendo, non si può trarre vantaggio da questo tipo di conoscenza per migliorare le prestazioni del sistema. Un agente esperto sarà più veloce di un neo assunto, sia durante che al termine della telefonata, in fase di registrazione dell'esito.

E' evidente che il modello di Samuelson porta con sé anche i problemi legati alle tecnologie degli anni '80. Per quanto riguarda il limite del numero delle linee telefoniche a disposizione, ad esempio, con Samuelson è un vincolo fondamentale. Oggi, grazie alle nuove tecnologie VoIP, questo limite non esiste più. Il numero di linee fisiche, a disposizione del Call Center, rimane comunque limitato, ma non lo è quello delle linee virtuali. Anche il numero di agenti era un limite importantissimo, che oggi è stato superato grazie agli Interactive Voice Response (IVR). Non solo la linea viene virtualizzata, ma anche l'agente stesso.
Un altro importante limite, dovuto agli anni che questo modello ha sulle spalle, è sicuramente la minore potenza dell'hardware rispetto ai giorni odierni.

Negli anni ’80, il server dei Call Center aveva una potenza di calcolo ridicola rispetto alle macchine attuali. I vari componenti, come il dialer ed il sistema CTI, erano certamente meno veloci di quelli attuali.

Grazie alle nuove tecnologie, il modello proposto gode di una maggiore flessibilità rispetto a quello di Samuelson. Come detto in precedenza, grazie al VoIP e agli IVR, il numero di linee e di agenti non è più un limite.
Il modello è stato basato sull’utilizzo di entità ed attributi relativi. L’entità agente, ad esempio, ha come attributi il tempo di inzio telefonata e la campagna su cui sta lavorando. L’entità Call Center ha, ad esempio, come attributi, il numero delle linee ed il numero di agenti. La telefonata può avere come attributo la tipologia e la predizione sulla durata. Questo tipo di struttura ad attributi, è stata scelta per rendere il modello pronto a qualsiasi cambiamento. In base al contesto o a nuove esigenze è sempre possibile aggiungere nuovi attributi per considerare anche gli altri fattori in gioco.
Ad esempio, grazie a questo meccanismo, è possibile gestire la possibilità di avere più campagne in esecuzione, nello stesso momento. Basta assegnare i giusti valori agli attributi delle entià che si vuol far interagire.

Di sicuro, Samuelson ha un vantaggio dovuto all’esperienza sul campo e soprattutto ad anni di prove e di utilizzo del sistema. Lui stesso, afferma nel suo articolo, che all’inizio aveva scelto alcuni indicatori, utili per la predizione, che, in seguito a numerose prove, si sono rivelati inutili. Allo stesso tempo, però, ne ha individuati altri che gli hanno permesso di migliorare, non di poco, le prestazioni del suo algoritmo di predizione.

Capitolo 4

Implementazione del modello

4.1 Tecnologie utilizzate

Prima di poter implementare il modello, è stato necessario individuare il server PBX da utilizzare, per gestire tutte le chiamate in maniera automatica.
A tale scopo, è stato scelto Asterisk, perché è una soluzione open source, quindi gratuita. Un altro buon motivo per scegliere questo PBX, è il fatto che gode di un ottimo supporto on line, con tanta documentazione ed esempi vari. Inoltre, dovrebbe essere una soluzione valida ed affidabile, poiché sono numerose le aziende che lo utilizzano.
L'ultima versione di Asterisk, è scaricabile gratuitamente dal sito ufficiale della piattaforma, all'indirizzo *http://www.asterisk.org/downloads*.
In questo caso, è stata installata la versione 1.6, su un sistema Linux Ubuntu 8.04.
Per interagire con il server, tramite le apposite Manager API, è stata scelta una libreria open source, chiamata Asterisk.Net e scaricabile al seguente indirizzo: *http://sourceforge.net/projects/asterisk-dotnet/.*
Tramite questa libreria, è possibile interagire con il server sfruttando il paradigma di programmazione ad oggetti del linguaggio C#.Net/Mono. Per esigenze pratiche, spiegate più avanti nel capitolo, è stato necessario sviluppare un ulteriore strato di astrazione basato su Asterisk.Net. E' stata sviluppata un'altra libreria, sempre in C#.Net/Mono, chiamata EasyAsteriskInterface. Lo scopo di quest'ultima è di rendere immediata e semplice l'interazione finale con il server.

L'implementazione del modello che viene proposta in questo capitolo, è solo un possibile esempio concreto di prototipo funzionante.
In realtà, il modello proposto può essere implementato con qualsiasi linguaggio supportato dalle API esposte dal server. La logica del modello è completamente indipendente dal tipo di linguaggio di programmazione utilizzato.

4.1.1 Installazione e configurazione di Asterisk

Dopo aver scaricato e scompattato il pacchetto relativo, l'installazione è pressocché automatica.

Nella cartella appena scompattata, i comandi da eseguire nella console sono:

make clean
make install

Al termine, verrà stampato un messaggio che avvisa che il server PBX Asterisk è stato correttamente installato.
Per avviarlo, è necessario, da utente root, digitare nella console:

asterisk -cv

Da questo momento in poi, il server è raggiungibile all'indirizzo della macchina su cui è installato ed è in ascolto, di default, sulla porta 5038.
Asterisk è fornito anche di una comoda interfaccia web ma non è inclusa direttamente nel pacchetto di installazione del server. Per scaricare la GUI è necessario avere un client di *Subversion (SVN).* In questo caso è stato utilizzato un client scaricabile all'indirizzo *http://www.nongnu.org/cvs/* .
Tramite questo programma, è possibile scaricare l'ultima versione della GUI di Asterisk, che è presente all'indirizzo *http://svn.digium.com/svn/asterisk-gui/trunk asterisk-gui*.

Prima ancora della GUI, va installato un Web Server[1], poiché l'interfaccia è una applicazione web a tutti gli effetti. Nel caso in questione, è stato installato Apache, scaricabile all'indirizzo http://www.apache.org/dyn/closer.cgi.
L'installazione della GUI è un pò meno facile ed intuitiva rispetto a quella del server.
Una volta scaricato e scompattato il pacchetto della GUI, digitare i seguenti comandi nella console della cartella appena decompressa :
configure
make
make install
make checkconfig

Al termine dell'esecuzione di questi quattro script, anche la GUI è stata installata. Eventuali problemi saranno riportati nella console.
Per adesso l'interfaccia web non è ancora utilizzabile, Asterisk deve essere prima configurato per poter essere usato via web.
Per far ciò, da utente root, modificare i seguenti file :

1. 'manager.conf'

   enabled = yes
   webenabled = yes
   bindaddress = 0.0.0.0

   Aggiungere anche la seguente sezione :

   [administrator]
   secret = wrxiur (inserire qui la password dell'utente administrator)
   read = system,call,log,verbose,command,agent,user,config
   write = system,call,log,verbose,command,agent,user,config

[1] Un server web è un programma, che si occupa di fornire, su richiesta del browser, una pagina web. Le informazioni inviate dal web server viaggiano in rete trasportate dal protocollo HTTP.

2. ‘http.conf’

enabled=yes
enablestatic=yes
bindaddress = 0.0.0.0

Dopo aver apportato le suddette modifiche, ricaricare le impostazioni di Asterisk digitando in console i comandi :

asterisk –cv
reload

A questo punto, il server è pronto per essere utilizzato dall’interfaccia web. Andando all’indirizzo *http://indirizzo_server:8088/asterisk/static/config/cfgbasic.html* si avrà accesso alla GUI di Asterisk. Attraverso questo strumento, sarà possibile registrare nuovi agenti, nuovi canali e nuove code, creare e modificare nuovi dialplan, oltre a gestire tanti altri aspetti legati al server PBX.

4.1.2 La libreria Asterisk.Net

E’ una libreria scritta in C#.Net ed compatibile anche con Mono, quindi può funzionare sia in ambiente Windows che Linux. Questa libreria è stata scelta perché è l’unica attualemente, liberamente scaricabile, che utilizzi un linguaggio ad oggetti. L’attuale versione 1.6.1.0, supporta fino alla versione 1.4 di Asterisk, per questo motivo sono state necessarie delle modifiche per renderla compatibile con la versione 1.6.
Prima di introdurre le modifiche apportate, di seguito sarà riportata la sua struttura e sarà spiegato il suo macro funzionamento.

La prima caratteristica da sottolineare è che si tratta di una libreria *event-driven*[1].

[1] La programmazione ad eventi è un paradigma di programmazione in cui il flusso del programma è largamente determinato dal verificarsi di eventi esterni.

L'interazione con Asterisk.Net è basata sulla gestione degli eventi scatenati, in base alla richiesta di esecuzione di una particolare azione. Permette l'interazione con Asterisk sia tramite API AGI che Manager. Per comunicare con il server, utilizza il protocollo TPC/IP e, tramite appositi socket, invia e riceve stringhe.
Per richiedere l'esecuzione delle azioni al server, formatta in maniera adeguata la stringa di richiesta, nella forma *"chiave:valoreCRLF"*(vedi cap. 1), ed invia lo stream di caratteri, sfruttando il socket in uscita verso il PBX. Quando il server riceve la richiesta di esecuzione di una azione, dopo averne controllato la validità, la esegue, ed invia al client o una semplice risposta con l'esito, oppure un evento ed i relativi argomenti. Anche in questo caso, il server utilizza lo stesso socket e lo stesso meccanismo, basato su stream di caratteri, per comunicare con il client.
Quando riceve una risposta o un evento, Asterisk.Net parsa il pacchetto ed, in base al tipo, crea un nuovo oggetto Event o Response.
Tramite questo meccanismo, è possibile sapere sul client cosa accade sul server, ed è possibile intervenire da remoto sul comportamento del server stesso.
Per essere completamente conforme all'interfaccia del server, la libreria ha tante classi quante sono le azioni, gli eventi e le risposte che vengono esposte come API.
In questo modo, ogni singola attività del server può essere mappata su un oggetto sul client.

Tralasciando l'implementazione delle API AGI, poiché non sono mai state utilizzate durante tutto il lavoro, è importante capire come funziona la gestione delle API Manager. Con il metodo *Login(),* dell'oggetto *ManagerConnection,* è possibile loggare l'utente, con diritti di amministratore, sul server. In base agli eventi che servono, tramite ManagerConnection, si possono registrare i relativi handler e, quindi, gestire sul client eventi scatenati in remoto.
L'oggetto *ManagerReader* ha il compito di svolgere il lavoro di basso livello, di scrittura e lettura dal socket di comunicazione con il server. Sfruttando i metodi e le proprietà di oggetti minori, crea il socket, formatta la stringa da inviare, in base al formato standard delle specifiche, e la invia. Quando, invece, riceve una stringa dal server, la parsa e, in base al valore della chiave, crea l'oggetto adeguato all'informazione appena ricevuta.

Per inviare una azione al server, viene utilizzato il metodo *SendAction(),* sempre del ManagerConnection. Quest'ultimo, come argomento, accetta l'oggetto relativo all'azione da mandare. La libreria offre anche altre classi, non direttamente collegate all'attività sul server, ma, piuttosto, a quella del client. Con la classe *Logger* è possibile loggare su file testuali, tutto ciò che avviene sul client, azioni richieste, eventi e risposte ricevute dal server. Esistono anche classi per gestire la cifratura dei messaggi scambiati tra client e server, oltre a classi che facilitano l'utilizzo dei pool di thread, con la gestione delle risposte asincrone dal server.

A causa di alcuni bachi, e poiché l'ultima versione di Asterisk.Net non supporta Asterisk 1.6, è stato necessario apportare delle modifiche al codice sorgente della libreria.
Nella nuova versione di Asterisk, è stato introdotto un nuovo evento, chiamato *Bridge*, che viene scatenato quando inizia una nuova chiamata tra agente e cliente.
La versione di Asterisk.Net non supporta ancora questo evento, e tutte le volte che viene scatenato dal server ed arriva al client, la libreria solleva una eccezione non gestita.
Questo provoca una disconnessione dell'utente dal server e, quindi, non è più possibile ricevere ed inviare dati, finché non viene riconnesso l'utente.
Per ovviare a questo problema, è stato necessario modificare il codice sorgente della classe ManagerReader. Alla linea numero 322, è stato aggiunto un controllo per verificare il tipo di evento ricevuto dal server. Se si tratta dell'evento Bridge, allora l'evento viene scartato e non utilizzato sul client.
Di seguito la parte di codice in questione :

```
if (packet.ContainsKey("event")) {
    if (!packet["event"].Equals("Bridge"))
        mrConnector.DispatchEvent(packet);
}
```

Un problema simile si è verificato sulla gestione degli attributi delle code. Nell'ultima versione di Asterisk ,è stata aggiunta l'opportunità di avere l'attributo *InternalActionId* settatto ad un valore nullo, anche all'oggetto coda. Non prevedendo questa opportunità, l'attuale libreria Asterisk.Net, scatena una eccezione non gestita, con le stesse

conseguenze del caso precedente. Così come prima, è stato necessario introdurre un controllo sul valore dell'attributo, per risolvere il problema.

Alla riga 2037, della classe ManagerConnection, è stata apportata la seguente modifica :

```
if (e is ResponseEvent)
{
     ResponseEvent responseEvent = (ResponseEvent)e;
     if (!(responseEvent.InternalActionId == null)) {
         ResponseEventHandler eventHandler =
(ResponseEventHandler)GetResponseEventHandler(responseEvent
.InternalActionId.GetHashCode());
[...]
```

Con le suddette modifiche apportate, la versione 1.6.0.1 di Asterisk.Net è completamente compatibile e funzionante con l'ultima versione 1.6 del PBX Asterisk.

Con il meccanismo event-driven che viene usato dalla libreria, non sempre è facile reperire informazioni. Ad esempio, per ottenere dei dati su un agente o su una coda, bisogna inviare delle azioni al server e, successivamente, per collezionare le informazioni di cui si ha bisogno, si devono gestire almeno due eventi diversi per ogni richiesta. E' un meccanismo scomodo quando, per ragioni pratiche, di logica dell'algoritmo, bisogna richiedere spesso queste informazioni e c'è la necessità di gestire diverse strutture dati per ogni evento.
Per questo motivo, durante la fase di implementazione del modello, è stata sviluppata un'altra libreria, chiamata EasyAsteriskInterface (EAI)[1].

[1] La libreria EAI non è un progetto pubblico. E' stata sviluppata, partendo da zero, durante il lavoro di tesi.

4.1.3 La libreria EasyAsteriskInterface (EAI)

E' stata sviluppata in C#.Net/Mono e si basa sulla libreria Asterisk.Net. In realtà, è una libreria wrapper, che si occupa di gestire, in maniera trasparente all'utilizzatore, tutto il meccanismo degli eventi introdotto in precedenza.

La EAI è composta da tre classi :

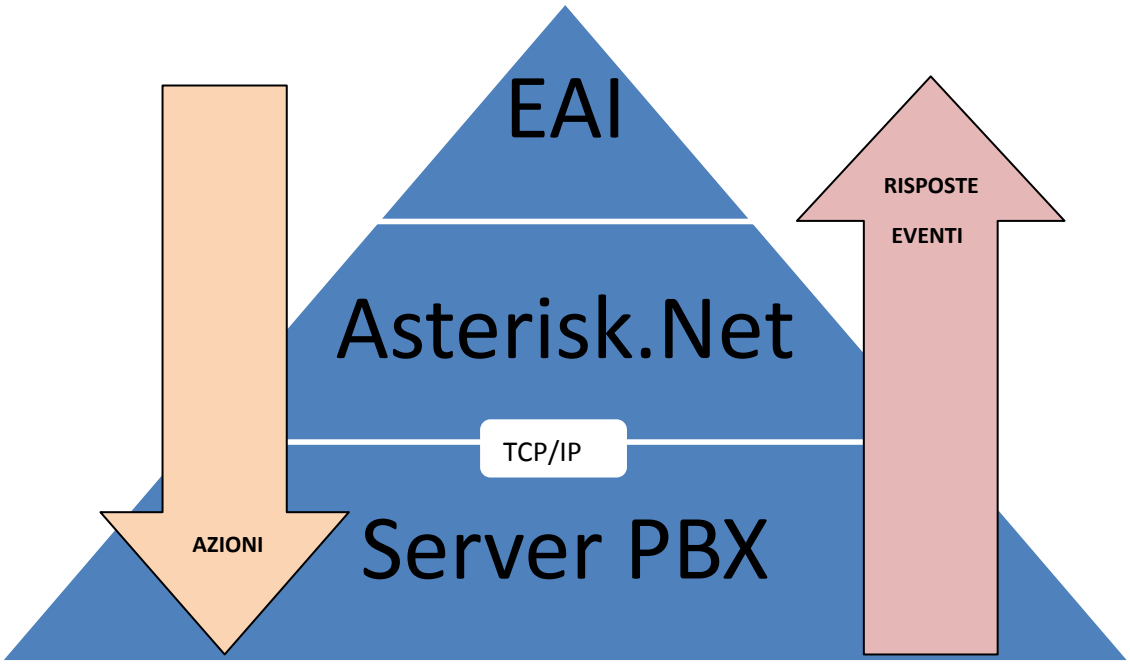
- EAIMain
- Agent
- CallInfo

La prima, come si può intuire anche dal nome, è la classe principale. Si occupa di interfacciarsi direttamente con lo strato sottostante, ovvero Asterisk.Net.
Espone il metodo *CreateManager()* che restituisce un riferimento ad un oggetto ManagerConnection. Attraverso il metodo *ManagerAddEvents()*, permette di registrare gli hanlder, relativi agli eventi che può gestire ManagerConnection.
Il metodo più importante è sicuramente *getAgents()* che restituisce una lista di oggetti Agent. E' questo metodo che racchiude tutta la logica di gestione degli eventi interni, per ottenere tutte le infomazioni sugli agenti presenti nel server PBX.
Gli altri metodi, *ExecuteLogIn()*, *ExecuteLogOff()* e *ExecutePause()*, servono per effettuare il login, logout e la pausa di un agente.
Inoltre, la libreria EAI, permette di rilanciare tutti gli eventi ricevuti dal server. Anche in questo caso, basta semplicemente registrare gli handler relativi, per poter gestire, dal livello più alto, gli eventi scatenati dallo strato inferiore.

La classe Agent serve a rappresentare un agente del sistema PBX, con tutte le sue proprietà importanti come: il suo user, la sua password, le code a cui è associato, lo stato attuale.

Lo stesso vale per CallInfo, rappresenta una chiamata effettuata. La classe ha proprietà molto importanti, quali: identificativo agente chiamato, tempo di inizo e fine chiamata, il canale su cui è stata effettuata, la coda a cui è stata smistata.

Di seguito lo schema di come interagiscono tra loro le librerie.



(Figura 7)

Per capire perché è stato davvero necessario implementare la libreria EAI, di seguito è stato riportato un esempio di codice in C#, per ottenere tutte le informazioni che si possono avere dal server sugli agenti :

```
EAIMain eai = new EAIMain();
ManagerConnection manager = eai.CreateManager();
eai.ManagerAddEvents(manager);
manager.Login();
List<Agent> agents = eai.getAgents();
```

Per ottenere lo stesso risultato, usando solo la libreria Asterisk.Net, occorrono 10 volte di più come quantità di righe di codice, senza contare la gestione di due diversi eventi.

4.2 Struttura logica del modulo software

Il modulo software, che implementa i due modelli proposti, per risolvere il problema del Predictive Dialing, è stato chiamato *Crystal*.
E' stato sviluppato con il linguaggio C#.Net e interagisce con un database Microsoft SQLServer 2008 Express.

Il database è fondamentale per questa applicazione, perché permette di memorizzare i dati, su ogni agente e per ogni campagna, utili per l'esecuzione corretta dell'algoritmo di predizione. Lo schema del database è composto da tre tabelle : T_Agents, T_Campaign e T_CampaignData.
T_Agents è una tabella che ha un solo campo, di tipo chiave, e memorizza l'id dell'agente all'interno del sistema PBX.

T_Campaign serve a memorizzare dati relativi ad ogni campagna pubblicitaria, è composta da sei campi :

- IDentificativo univoco nel database
- CampaignExtension (identificativo di una campagna pubblicitaria nel PBX)
- Average (durata media della telefonata per questa campagna)
- StandardDeviation (deviazione standard per questa campagna)
- Variance (varianza standard per questa campagna)
- Agent_ID (chiave esterna per la relazione con l'agente)

T_CampaignData serve a memorizzare i dati relativi ad ogni chiamata, è composta :

- IDentificativo univoco nel database
- StartedTime (istante di inizio chiamata)
- EndedTime (istante di fine chiamata)
- Campaign_ID (chiave esterna per la relazione con la campagna pubblicitaria)

Con questa struttura del database, è possibile tener traccia di ogni chiamata, per ogni campagna pubblicitaria, per ogni agente.

E' stata sviluppata una classe apposita, chiamata *DAL*, come punto di centralizzazione per interagire con il database. E' composta da 7 metodi statici :

- InsertNewAgent (inserisce un nuovo agente nel database)
- InsertNewCampaign (inserisce una nuova campagna nel database)
- InsertCampaignData (inserisce nuovi dati di una campagna nel database)
- UpdateCampaignData (aggiorna i dati di una campagna)
- GetCampaignID (recupera i dati di una campagna dall'ID e dall'agente)
- GetAgentCampaigns (recupera i dati di una campagna dall'agente)
- GetCallProperties (recupera i dati di una campagna dalla campagna e dall'agente)

Con questi metodi è possibile, aggiungere nuovi elementi nel DB, aggiornare solo le informazioni che lo necessitano, recuperare i dati di particolari campagne o di tutte quelle a cui un agente partecipa.

Per mantenere sincronizzati il database del modulo con il database del PBX, è necessario richiamare il metodo SetUPDataBase(), della classe Utils.
Questo metodo ha il compito di leggere dal server PBX la lista completa di tutti gli agenti esistenti, e di sincronizzarla con quelli presenti nel proprio database.
In questo mdo, è sempre possibile tener traccia dell'attività di tutti gli agenti, anche di quelli aggiunti successivamente alla prima installazione.

Ogni volta che il modulo viene avviato, legge tutti i dati dal database e popola apposite strutture dati, che saranno poi utilizzate durante tutta l'esecuzione del programma.
I principali oggetti, che vengono utlizzati per tale scopo, sono :

- CallProperties
- Campaign
- AgentDump

Tutti i dati che riguardano una singola telefonata, vengono letti dal database ed inseriti nell'oggetto CallProperties. Le sue proprietà sono esattamente quelle della relativa tabella. L'id univoco della chiamata, il tempo di inizio e fine, ed, in più, anche la durata, ottenuta come semplice differenza tra fine ed inzio.
Ogni campagna presente nel database, viene mappata su un oggetto Campaign.
Come nel caso precedente, le proprietà dell'oggetto sono le stesse della relativa tabella, ma, in più, ha anche un riferimento ad una lista di oggetti CallProperties.
Come è ovvio immaginare, questa lista è riempita con tutte le telefonate che sono state effettuate per la campagna a cui l'oggetto si riferisce.
Per chiudere il cerchio, ogni campagna è collegata ad un agente, tramite l'oggetto AgentDump. Quest'ultimo, ha solo due proprietà, un riferimento ad un oggetto agente e ad una lista di oggetti Campaign, poiché un singolo agente può partecipare a diverse campagne pubblicitarie.

La prima cosa che viene svolta dal modulo, è la creazione di un oggetto ManagerConnection tramite la libreria EAI. In seconda istanza, viene richiamato il, già discusso, metodo getAgents. Subito dopo aver ricevuto dal server, la lista di tutti gli agenti presenti nel sistema, viene richiamato un altro importante metodo statico della classe Utils, *GetAgentDumpList()*.
Quest'ultimo, è un metodo, che permette di recuperare, in una unica soluzione, tutti gli agenti, con tutte le relative campagne e tutte le relative chiamate.

```
static public List<AgentDump> GetAgentDumpList(List<Agent>
agents){

            List<AgentDump> dump = new List<AgentDump>();

            foreach (Agent agent in agents)
            {
                int ext = agent.Extension;
```

```
                List<Campaign> campaigns =
DAL.GetAgentCampaigns(ext);

                foreach (Campaign campaign in campaigns)
                    campaign.Calls =
DAL.GetCallProperties(ext, campaign.DbID);

                AgentCampaignsList campaignList = new
AgentCampaignsList(campaigns);
                KeyValuePair<int, AgentCampaignsList>
agentCampaignsList = new KeyValuePair<int,
AgentCampaignsList>(ext, campaignList);

                AgentDump singleDump = new AgentDump(agent,
agentCampaignsList);

                dump.Add(singleDump);
            }
            return dump;
        }
```

E' un doppio ciclo innestato. Nel primo, vengono recuperate le campagne pubblicitarie dell'agente, nel secondo le chiamate di ogni campagna. Di volta in volta, vengono creati oggetti appositi, che rappresentano le varie entità in gioco con le rispettive proprietà.
Al termine di ogni ciclo, i vari oggeti creati vengono inseriti in specifiche liste per essere recuperati ed utilizzati in seguito.

Il modulo ha anche il compito di caricare i numeri telefonici dei clienti da chiamare. Per questo scopo, può interagire direttamente con un CMR, oppure prelevare i numeri da altre fonti.

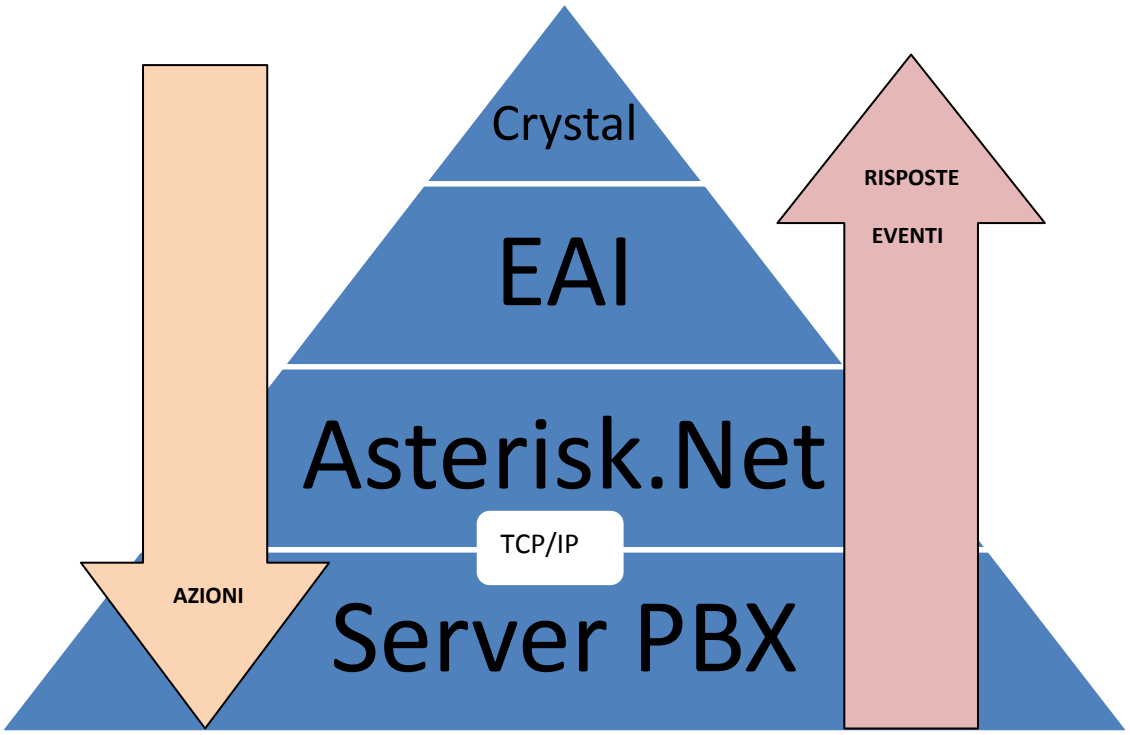
Avendo a disposizione tutti i dati necessari, il modulo è pronto per inziare ad effettuare il primo giro di telefonate. Sin dalla prima chiamata del giorno, viene utilizzato l'algoritmo predittivo, per individuare il numero di chiamate da effettuare.
Grazie al metodo *CalculateNumberOfCalls*, che prende come argomento il tipo di modello da utilizzare, è possibile ottenere il numero K di chiamate da effettuare.
A questo punto, viene chiamato il metodo *ExecuteCalls,* che manda al server, K volte, il comando per effettuare una telefonata verso l'esterno.
Quando una o più telefonate iniziano, dal server vengono scatenati tutti gli eventi necessari per monitorare lo stato di avanzamento delle chiamte, degli agenti e delle code. E' proprio sfruttando questi eventi che è possibile capire quando un cliente ha risposto e, quindi, si può metterlo in contatto con il primo agente libero.
Allo stesso modo, vengono gestite tutte le altre chiamate. Quelle non risposte, risultate occupate, libere o fax, vengono comunque inserite nel database del modulo per aumentare i dati a disposizione dell'algortimo predittivo.
Anche le chiamate risposte vengono registrate, ed il sistema tiene traccia soprattutto della loro durata.
L'algoritmo predittivo, viene applicato ogni volta che bisogna effettuare un nuovo turno di telefonate. Il tempo che intercorre tra un turno ed il successivo, varia in base alla durata media delle telefonate, scelte all'interno di un intervallo valido, secondo i criteri di chi installa e configura il modulo.
La gestione delle chiamate, ovvero, la composizione del numero, il controllo dello stato della linea, lo smistamento delle telefonate, viene effettuata dal server PBX.
Il modulo sviluppato, ha solo il compito di dettare i tempi per compiere tali azioni, quando chiamare, quando smistare le telefonate e quando chiudere.
Nel caso in cui, arrivino più chiamate, di quanti sono gli agenti liberi in quel momento, il modulo invia al server l'azione di *HangUp* per chiudere la telefonata.
Questi sono i casi in cui si verifica il fenomeno delle silent call, già discusso in precedenza.
In base alla logica dei due modelli predittivi proposti, più dati si hanno a disposizione per la valutazione degli indicatori percentuali, e più il risultato della predizione sarà affidabile. La situazione ideale, sarebbe avere a disposizione, già dal primo utilizzo del modulo, dati di campagne pubblicitarie effettuate in precedenza.

Se questo non fosse possibile, è stato previsto una doppia modalità di funzionamento del sistema, *Training e Working mode*. La prima modalità, imposta un comportamento passivo del modulo, che ha solo il compito di monitorare lo stato e gli esiti di tutte le chiamate, registrando nel proprio database le informazioni utili. Durante questa fase, il sistema funziona con un meccanismo simile a quello basato su ratio.
Non avendo ancora abbastanza dati con cui effettuare la predizione, ad ogni turno di chiamate, vengono contati quanti sono gli agenti effettivamente liberi. Il valore ottenuto viene moltiplicato per un certo fattore, tale che, il risultato della moltiplicazione, non superi mai il numero massimo di linee che il Call Center vuole utilizzare.
In realtà, durante l'utilizzo del modulo in modalità Training, è possibile utilizzare un qualsiasi altro sistema automatico, per poter decidere quando e quante chiamate effettuare. Perché, in ogni caso, al suo avvio, il modulo registra comunque gli handler degli eventi di interesse, ed il server invia sempre i dati richiesti.
Esiste un file di configurazione dell'applicazione, *Crystal.config*, in cui vengono riportati i valori di alcune importanti proprietà, utili per il corretto settaggio e funzionamento del programma stesso.

Nel suddetto file, vengono memorizzati :

- la stringa di connessione al database
- l'indirizzo del server PBX Asterisk a cui collegarsi
- la porta remota del server
- le credenziali per l'account di amministratore
- il tipo di modalità di funzionamento (0 = Training, 1 = Working)
- il valore di ratio da usare, eventualmente, durante la modalità di training
- il modello da usare (1= distribuzione normale, 2 = normale inversa)

Il nuovo, e definitivo, schema di interazione è il seguente :



(Figura 8)

Il progetto Crystal è l'ultimo tassello che mancava per poter avere un server PBX Asterisk, che sfrutta un modulo di Predictive Dialing scritto in C#.Net che implementa i due modelli proposti.

4.3 L'algoritmo principale

Come anticipato anche in precedenza, il modulo supporta entrambi i modelli proposti.
Ciò che li differenzia, è la prima parte, relativa alla funzione f_1, per il calcolo del numero di agenti a disposizione, in un dato istante t_s.
Per questo motivo, gran parte dell'algoritmo, è valido per entrambi i modelli, le variazioni, effettuate in base al modello scelto, sono davvero minime.

Partendo dal template proposto nel secondo capitolo, una possibile implementazione concreta, in C#.Net, è la seguente :

```
int totalCalls,freeAgents , interval;
DateTime timeToCall = DateTime.Now;
interval = CalculateRecallTime();
timer.interval = interval;

while(isOn){

     freeAgents= CalculateFreeAgentsNumber(timeToCall);

     if(freeAgents > 0){
          totalCalls = CalculateNumberOfCalls(freeAgents);
          timer.Start();
     }

     OnTimerElapsed{
          timer.Stop();
          ExecuteCalls(totalCalls);
          timeToCall.AddSeconds(interval);
     }

     OnEndCalls{
          isOn = false;
     }
     }
```

Per motivi di facilità di comprensione, sono state riportate solo le parti importanti dell'algoritmo. E' stata omessa, ad esempio, la gestione degli errori e di altri controlli per valutare il flusso dell'applicazione. Anche la sintassi usata, quella di C#.Net, non è del tutto completa, mancano alcune dichiarazioni e le firme dei metodi sono state semplificate.

Considerando il sistema, già a regime ed in modalità *Working*, ciò che viene effettuato è, nell'ordine :

1. Calcolato il tempo espresso in secondi, tra due turni di chiamate [*CalculateRecallTime()*]

2. Calcolato il numero di agenti liberi nel momento in cui verranno effettuate le nuove telefonate [*CalculateFreeAgentsNumber(timeToCall)*]

3. Calcolare il numero delle telefonate totali, da effettuare verso l'esterno, in base al valore valutato al punto precedente [*CalculateNumberOfCalls(freeAgents)*]

4. Al momento giusto, effettuare tutte le chiamate stabilite, e decidere quando effettuare le successive [*OnTimerElapsed*]

Il primo metodo, *CalculateRecallTime*, calcola la media delle durate, di un certo numero di telefonate, comprese all'interno di un preciso intervallo di durata.
Questo intervallo deve essere scelto in maniera adeguata, poiché quando effettuare la predizione è una decisione importante. Un valore troppo basso o troppo alto può portare a predizioni poco affidabili. Prendendo, ad esempio, la campagna pubblicitaria B (vedi cap. 3.3), dal grafico è evidente che il maggior numero numero di telefonate termina nell'intervallo [0,60] , anche fino a [0,100] sarebbe accettabile.
Avendo verificato che può essere un intervallo significativo, dal punto di vista del numero di telefonate che vi ricadono, è lecito sceglierlo per calcolare le durata media su tutte le telefonate comprese. Il valore risultante sarà usato come tempo divisorio tra due turni di telefonate. Dovendo, comuque, scegliere quando effettuare un nuovo turno di chiamate, è stata pensata questa soluzione, perché ritenuta una buona intuizione logica.
Se mediamente, la maggior parte delle chiamate, dura T secondi, allora è logico richiamare esattamente dopo lo stesso tempo T.

Il secondo metodo, *CalculateFreeAgentsNumber*, prende come argomento il tempo t_s, in cui si desidera effettuare le telefonate.

All'interno del suddetto metodo, vengono calcolati, sia gli agenti che sono sicuramente liberi subito, e sia quelli che si prevede lo siano. Il primo calcolo è sicuramente semplice, basta valutare lo stato di tutti gli agenti e contare quanti risultano liberi.
Per il secondo valore, invece, è leggermente più complicato.
Tutte le volte che un agente riceve una telefonata, ed inizia a parlare con un cliente, viene scatenato un particolare evento, *AgentConnect*. Lo stesso avviene quanto la chiamata termina, ma l'evento è *AgentComplete.*
In entrambi i casi, tramite l'argomento degli eventi, è possibile ottenere informazioni come tempo di inzio e fine chiamata, oltre che identificativo dell'agente e della campagna pubblicitaria. Nell'evento AgentConnect, oltre alla registrazione del tempo di inzio della telefonata, viene effettuata la predizione sulla durata della stessa.
Questo, ad esempio, è un punto di diversità tra i due modelli.
Nel caso del modello con distribuzione normale, come durata prevista della telefonata, si può assumere l'intervallo calcolato tramite la funzione densità di distribuzione.
A tale scopo, la classe Utils, espone quattro metodi statici:

- CalculateAverage
- CalculateVariance
- CalculateStandardDeviation
- CalculateGauss

Vengono usati per calcolare rispettivamente: media, varianza, deviazione standard e per applicare la soluzione tabellare della funzione di ripartizione della distribuzione normale.
Mentre, nel caso della distribuzione normale inversa, viene calcolato un numero a caso, che rispetti la funzione densità di distribuzione, e questo valore sarà considerato come potenziale durata delle chiamata. (vedi capitolo 3)
Con questo meccanismo, ad ogni agente occupato in una conversazione, è anche associato una previsione sulla durata delle telefonata. Valutando quest'ultima, in relazionea al tempo t_s, è possibile calcolare quanti agenti potrebbero essere liberi in quel dato istante.

Avendo a disposizione, il numero di agenti, sicuramente e potenzialmente liberi, basta effettuare la somma, per ottenere l'output del metodo CalculateFreeAgentsNumber.

Il terzo metodo, *CalculateNumberOfCalls,* accetta come argomento il numero previsto di agenti liberi. Il suo scopo è quello di implementare la funzione f_2, che è uguale per entrambi i modelli. Sulla base delle informazioni contenute nel database, vengono calcolate tutte le chiamate risultate: occupate, non risposte, in errore e fax/fuori servizio. Successivamente, per ogni totale ottenuto, ne calcola la percentuale risultante su tutte le chiamate effettuate. In pratica, esegue il calcolo degli indicatori negativi del modello.
Infine, dopo aver calcolato i valori numerici delle precedenti percentuali, rispetto al numero totale di agenti liberi, li somma proprio quest'ultimo.
Il risultato della somma è l'output del metodo, nonché il valore della predizione finale.
Come si evince, da quanto detto fino ad ora, la predizione finale è preceduta da un'altra parziale, che riguarda l'individuazione del potenziale numero di agenti liberi.

La parte finale dell'algoritmo, riguarda la comunicazione al server, di dover eseguire un certo numero di chiamate verso l'esterno, attraverso l'esecuzione del metodo *ExecuteCalls*. Sfruttando la libreria EAI, viene inviato al server, K (numero di chiamate da effettuare) volte il comando *OriginateAction*. Quest'ultimo accetta come argomenti, il numero da chiamare e l'identificativo del chiamante.
Successivamente, il modulo ha il compito di "ascoltare" e registrare gli eventi che il server genera, per ogni chiamata effettuata, e ricominciare dall'inizio tutte le azioni descritte in precedenza.

Capitolo 5

Implementazione di un simulatore e confronto pratico tra i modelli

5.1 Scopo del simulatore

Dopo aver formulato un modello risolutivo al problema del Predictive Dialing, averlo implementanto ed integrato con un server PBX, è stato sviluppato anche un simulatore per testare la bontà della soluzione proposta.

Il progetto del simulatore è stato nominato *CallCenterSimulator*, ed è stato sviluppato anch'esso con il linguaggio C#.Net.
Il simulatore è stato pensato per confrontare i modelli già presenti in letteratura con quelli proposti in questa sede. Tra gli unici due modelli già esistenti, l'unico, che è stato possibile simulare e testare, è quello basato sul fattore ratio.
Il modello di Samuelson, è stato descritto, dell'autore stesso, solo in maniera sommaria. Mancano i dettagli più importanti dell'algoritmo di predizione, per cui, è impossibile replicarlo e poterlo testare.
In commercio esistono vari software di simulazione per Call Center, ma sono tutti orientati alla simulazione del traffico inbound.
E' stato anche pensato di poter utilizzare software più generali, di simulazione di sistemi dinamici, come ad esempio iThink, PowerSim o Venism. Anche questa idea è stata da

subito scartata, perché sono tutti sistemi, oltre che a pagamento, complessi da imparare ad utilizzare in un tempo ristretto.
L'unica soluzione, quindi, è stata quella di scrivere da zero un simulatore per outbound Call Center.

5.2 Problemi affrontati

Durante la fase di studio ed ideazione del simulatore, i principali problemi che sono stati riscontrati sono :

1. Simulare gli esiti delle telefonate in maniera coerente con la realtà
2. Simulare l'avanzamento del tempo, senza far durare ogni simulazione quanto una giornata lavorativa reale
3. Individuare criteri validi per effettuare i confronti tra i diversi modelli
4. Assegnare un certo grado di affidabilità al simulatore stesso

Per ottenere una simulazione valida ed attendibile, è necessario riuscire a mantenere le stesse distribuzioni degli esiti delle chiamate reali. Non sarebbe corretto utilizzare una semplice funzione randomica per tale scopo. Ad esempio, nella realtà, le chiamate non risposte si concentrano maggiormente durante le ore del mattino, quelle occupate vengono distribuite, in manera quasi uniforme, su tutte le ore della gionata lavorativa. E' necessario, quindi, mantenere queste caratteristiche anche nella simulazione.
Per questo motivo, è stato deciso di prelevare gli esiti direttamente dai tre database a disposizione (vedi capitolo 3), riguardanti reali campagne di telemarketing, che si sono svolte nel mese di Luglio del 2008. L'esito di ogni telefonata, viene preso in ordine, dal primo all'ultimo, ed utilizzato dal simulatore come se fosse quello reale della telefonata appena effettuata.

Il secondo problema, riguarda come gestire l'avanzamento temporale in maniera corretta, ma allo stesso tempo non dipendente dallo scorrere dei minuti della realtà. Per questo tipo di simulazione, per tutti i motivi discussi nei precedenti capitoli, il fattore tempo è più che fondamentale. La prima soluzione, è stata quella di leggere

direttamente dai database reali i tempi di inizio e fine chiamata, ed utilizzarli come orologio interno al simulatore. Come risoluzione del problema, sarebbe stata eccellente. Purtroppo, però, nei database, è memorizzato anche l'istante di fine, solo per le telefonate risposte.

Per tutte le altre chiamate, risultate non risposte, occupate, fax o errori, l'unica informazione a disposizione è il tempo di inizio, ma non è presente quello di fine.
Poiché questo tipo di telefonate è, sicuramente, in numero molto maggiore rispetto a quelle realmente risposte dal cliente, non è stato possibile effettuare una approssimazione e considerare solo gli esiti positivi.
Allora, è stato pensato di assegnare ad ogni tipo di esito non positivo un certo valore temporale, espresso in secondi. Ad esempio, prendendo in esame le chiamate occupate, e decidendo di farle durare 5 secondi, si avrebbe che una chiamata inziata alle ore 11.00.00, si potrebbe considerare terminata alle 11.00.05.
Anche questo approccio non è stato utilizzato perché, non avendo dati reali in merito, non è stato possibile sapere quanto tempo assegnare ad ogni tipo di esito non positivo.
Inserire dei valori del tutto a caso, non avrebbe reso certamente il simulatore attendibile.
La soluzione che è stata utilizzata, come orologio interno al sistema, prevede di dividere il numero totale delle chiamate, presenti nel database reale, per il valore 720. Quest'ultimo è il numero esatto di minuti della giornata lavorativa del Call Center, che va dalle ore 9 del mattino alle ore 21 della sera. Un totale di 12 ore, ovvero 720 minuti, durante le quali vengono effettuate le chiamate verso i clienti.
Grazie a questa semplice operazione di divisione, è stato calcolato il numero *approssimativo* di chiamate, che il Call Center ha effettuato in un minuto, durante la campagna pubblicitaria a cui il database si riferisce.
Sfruttando questa informazione, è stato possibile ottenere un orologio interno al simulatore, che riesce ad approssimare il reale scorrere dei minuti.

Gli ultimi due problemi riscontrati, sono strettamente legati tra loro.
Nei reali sistemi di telemarketing, l'operato di un agente, non viene valutato solo in base a quanti e quali prodotti è riuscito a vendere, ma anche in base a quante telefonate ha partecipato in un'ora. I secondi totali di conversazione, a cui ha partecipato un

operatore,vengono divisi per il numero di ore di lavoro. Il risultato è indicativo della quantità di lavoro svolta da ogni agente.
Essendo un metodo di valutazione del tutto obiettivo e molto utilizzato, è stato scelto anche per il simualtore.

I risultati della simulazione dei tre modelli, sono stati confrontati usando come criterio il numero medio di secondi di conversazione effettuati in un'ora.

Per valutare l'affidabilità del simulatore, è stato pensato di utilizzare lo stesso criterio.
Calcolare sia sui dati reali che su quelli simulati, i minuti medi di conversazione per agente, per poterli confrontare.
Questa soluzione però non è stata utilizzata, perché non è noto l'algoritmo del modello, usato durante l'esecuzione delle campagne pubblicitarie, di cui si dispongono i dati.
Non è corretto confrontare valori ottenuti con algoritmi di cui non se ne conosce il funzionamento, sarebbe difficile interpretare i risultati.
L'unica strada percorribile, per cercare di capire se il simulatore è affidabile o meno, è valutare se i suoi risultati sono coerenti con quello che la teoria e la logica suggeriscono.
A parità di condizioni, ad eccezione del tipo di modello da usare, la teoria dice che il modello con ratio dovrebbe avere un numero più alto di silent call, ed un numero, uguale o minore, di secondi di conversazione per ora.

5.3 Struttura logica del simulatore e suo funzionamento

Le principali entià in gioco, di un reale outbound Call Center, sono :

- Gli agenti
- Le linee telefoniche
- Le code di agenti, una per ogni campagna pubblicitaria
- Le chiamate effettuate, con relativi esiti e durate
- Il modulo per il Predictive Dialing
- L'Automatic Call Distributor (ACD)

Anche per simulatore, è stato necessario introdurre le medesime entià. Per motivi di semplicità, è stato scelto di non prendere in considerazione né un possibile limite di linee telefoniche a disposizione, né più code nello stesso tempo, ovvero più campagne pubblicitarie eseguite in parallelo.
E' stato considerato un numero infinito di linee telefoniche ed una sola campagna pubblicitaria per volta.

Gli agenti sono rappresentati nel simulatore dalla classe Agent, che ha nove propretà :

- ID (identificativo univoco)
- Busy (valore booleano per indicare lo stato dell'agente. Se è True, allora l'agente è impegnato in una conversazione, altrimenti è su False)
- Start (indica il tempo di inizio chiamata)
- End (indica il tempo di fine chiamata)
- EndPrediction (indica la predizione sul tempo di durata della chiamata)
- TotalTalkTime (secondi complessivi spesi in comunicazione con i clienti)
- TotalTalkTimeByHour (secondi complessivi spesi in comunicazione con i clienti, in una ora)
- FirstCallReceived (booleano per indicare se l'agente ha effettuato la prima telefonata della giornata)

- FirstCallTime (l'istante in cui l'agente ha iniziato la prima chiamata della giornata

Il valore di Start viene recuperato in base al tempo segnato dall'orologio interno, quando inizia una telefonata con un cliente. Il valore End, invece, viene ottenuto sommando all'istante di inizio conversazione, la durata reale della chiamata, presa dal vero database con gli esiti.
EndPrediction è il risultato della predizione sulla possibile durata della telefonata. Viene calcolato in base al modello che si vuole simulare.
FirstCallReceived e FirstCallTime, servono a capire se, e quando, un agente ha già ricevuto la prima chiamata. Combinando queste informazioni con il tempo di inzio di ogni chiamata , è possibile sapere quante telefonate sono state effettuate nell'arco di un'ora da un singolo agente.

Per implementare l'entità *telefonata*, il simulatore utilizza la classe CallProperties, con cinque proprietà :

- CallType (esito della chiamata, risposto, libero, occupato)
- ID (identificativo univoco)
- Start (indica il tempo di inizio chiamata)
- End (indica il tempo di fine chiamata)
- Difference (durata della chiamata, ottenuta come differenza tra End e Start)

Come detto in precedenza, per mantenere una corretta distribuzione degli esiti, anche per le simulazioni, le telefonate vengo lette in ordine dal database e caricate in una lista di oggetti CallProperties.
Come è ovvio che sia, il modulo per il Predictive Dialing è quello implementato nel capitolo quattro, con i due modelli proposti, più l'implementazione del modello basato sul fattore ratio.
Per quanto riguarda l'ACD, viene semplicemente scelto il primo agente che ha la proprietà Busy impostata a false.

A questo punto è possibile introdurre in maggior dettaglio, la struttura del simulatore.

Al suo avvio, il simulatore carica tutte le chiamate dal database reale, e le inserisce in altrettanti oggetti CallProperties, impostandone le relative proprietà.
Di tutto questo, si occupa il metodo GetCalls(), di cui viene riportata la parte di codice più significativa :

```
for (int i = callsCounter; i < (callsCounter + number);
i++){
                CallProperties call = new CallProperties();
                call.DbID = results[i].ID;
                call.CallType =
(TypeOfCalls)results[i].Field2;
                call.Start = results[i].Field4;

                switch ((int)results[i].Field2){
                […]
                    case
(int)TypeOfCalls.Passatoadunoperatore:
                        call.End = results[i].Field5;
                        break;
                    default:
                        break;
                }

                call.Difference =
call.End.Subtract(call.Start).TotalSeconds;
                calls.Add(call);
            }
```

Come si può notare, è un ciclo all'interno del quale, viene letto ogni record del database delle chiamate. Di volta, in volta, vengono impostate le proprietà dell'oggetto CallProperies associato ed, infine, quest'ultimo viene aggiunto in una lista che conterrà

tutte le chiamate. In questa fase, è importante notare che la durata della chiamata risposta da un cliente, viene impostata sulla base di quella reale presente nel database.

Successivamente viene creata una lista di agenti, tutti inizialmente con le proprietà Busy,TotalTalkTime e FirstCallReceived, impostate rispettivamente a false,zero e false. In questo caso, il codice è molto semplice :

```
for (int i = 1; i <= total; i++){
      Agent agent = new Agent(i, false, 0,false);
      allAgents.Add(agent);
  }
```

Un'altra importante operazione, è quella di calcolare quante chiamate corrispondono ad un minuto dell' orologio interno al simulatore. Come già detto in precedenza, si tratta di una semplice divisione.

Una volta settato tutto l'ambiente di simulazione, è possibile inziare ad effettuare le telefonate. Tutto il cuore del simulatore è formato dal corpo del metodo ExecuteCalls().
E' un ciclo che continua finché ci sono ancora telefonate da effettuare, ovvero finché non vengono terminati tutti i record letti dal database inizialmente.
All'interno di questo ciclo, il primo passo è quello di calcolare quante telefonate effettuare. Per ottenere questo valore viene richiamato il metodo *CalculateNumberOfCalls,* già descritto nel capitolo precedente, che darà risultati diversi in base la modello che si vuole simulare.
Viene effettuato un controllo per verificare se il valore ottenuto è maggiore di zero o meno, per capire se bisogna uscire dal ciclo oppure no.
Se il valore K ottenuto è maggiore di zero, allora è possibile simulare le chiamate.
In questo ambiente,l'effettuazione di una telefonata, viene simulata prelevando dalla lista di tutte le chiamate, le K che servono. A questo punto, per ognuna, ne viene valutato l'esito. In caso di numeri occupati, fax o senza risposta, non viene effettuata nessuna operazione particolare, se non loggare questi eventi.

Se, invece, la telefonata risulta risposta dal cliente, si prende, dalla lista degli agenti, il primo che risulta libero.
Sia *agent* il riferimento al primo agento libero :

[...]

```
agent.Busy = true;
agent.Start = actualTime;
agent.End = agent.Start.AddSeconds(call.Difference);
agent.TotalTalkTime +=
agent.End.Subtract(agent.Start).TotalSeconds;

if (!agent.FirstCallReceived)
   agent.FirstCallTime = agent.Start;

if (agent.End.Subtract(agent.FirstCallTime).TotalHours < 1)
   agent.TotalTalkTimeByHour = agent.TotalTalkTime;

if (MODEL == 1)
   agent.EndPrediction =
agent.Start.AddSeconds(upInterval);


if (MODEL == 2){
   int rndDuration =
(int)randomNIG.ElementAt(random.Next(randomNIG.Count));
   agent.EndPrediction =
agent.Start.AddSeconds(rndDuration);
}

agent.FirstCallReceived = true;
```

[...]

Dal codice è chiara la logica utilizzata.

Lo stato dell'agente viene impostato ad occupato, vengono, anche, settati i tempi di inizio e fine conversazione. Oltre ad altre proprietà minori.

E' evidente come la predizione cambi in base al tipo di modello attualmente simulato. A tal proposito, si ricorda che il simulatore supporta anche il modello basato su ratio, che non richiede alcuna predizione.

Nel caso in cui non ci fossero agenti disponibili, la chiamata sarebbe calcolata come silent call. Il cliente è stato chiamato, ha risposto, ma nessun agente è libero per accettare la telefonata. Ogni volta che viene preso in esame un record del database, per simulare una nuova telefonata, viene anche incrementato un contatore che tiene traccia di quante chiamate sono state effettuate fino a quel momento.

Quando il valore del contatore diviene uguale al numero di chiamate per minuto, allora il tempo dell'orologio interno, *actualTime*, viene spostato in avanti di un minuto.

```
foreach (CallProperties call in actualCalls)
{
     k++;
     if (k == watch)     {
       actualTime = actualTime.AddMinutes(1);
       k = 0;
     }[...]
```

Così come descritto nel capitolo precedente, circa l'implementazione dei modelli, anche il simulatore utilizza gli stessi identici metodi per calcolare il numero di agenti liberi, il numero di chiamate da effettuare ed il momento esatto per eseguirle. Il meccanismo delle predizioni rimane,quindi, invariato.

Nel Call Center reale, però, quando un agente termina la telefonata, diventa automaticamente libero e rimesso in coda per accettare nuovi clienti.

Anche questo aspetto necessita di essere simulato.

Grazie al, già noto, metodo CalculateRecallTime, il simulatore sa quando effettuare un nuovo turno di chiamate.

Così controlla quanti e quali agenti attualmente impegnati, possono considerarsi liberi al momento richiesto :

```
int total = 0;
foreach (Agent agent in allAgents)
{
    if (agent.Busy == true)
    {
        if (agent.End.CompareTo(timeCall) < 0)
        {
            agent.Busy = false;
            total++;
         }
     }
}
return total;
```

Nel capitolo precedente, quando è stato fatto rifemento al modello basato sulla distribuzione normale inversa, è stato detto, semplicemente, che venivano generati dei numeri random secondo la funzione densità di probabilità.

Per la simulazione, è stato utilizzato il programma Matlab[1] per la generazione di 500.000 numeri random secondo la particolare distribuzione. La funzione utilizzata, *nigrnd*, non rientra in quelle standard, incluse con il programma, ma è stata reperita all'indirizzo http://www.allianz.de.

Il suo autore è il Dr. Ralf Werner[2].

L'output dell'esecuzione di questa funzione, è stato copiato in un file di testo, che fa parte del progetto del simulatore, e contiene una colonna con tutti i 500.000 numeri auto generati.

---

[1] MATLAB (abbreviazione di Matrix Laboratory) è un ambiente per il calcolo numerico e l'analisi statistica che comprende anche l'omonimo linguaggio di programmazione creato dalla MathWorks.

[2] http://www.mathworks.com/matlabcentral/fileexchange/authors/15753

Quando il simulatore deve utilizzare il modello 2, ovvero quello basato sulla distribuzione normale inversa, carica all'avvio, tutti i numeri in una lista apposita.
Al momento della predizione, ogni volta, viene scelto, in maniera randomica, uno tra tutti i valori caricati.
Il simulatore è una applicazione di tipo console, per essere lanciato necessita di quattro parametri, nell'ordine sono:

- Tipo di modello da simulare (0 = ratio, 1 = normale, 2 = normale inversa)
- Numero totale di agenti
- Fattore ratio
- Limite superiore dell'intervallo che parte da zero, per decidere ogni quanto effettuare un nuovo turno di chiamate

Ad esempio:

1. CallCenterSimulator.exe 0 15 1,5 300

Lancia il simulatore per valutare il modello basato su ratio, con 15 agenti, un fattore di 1,5 e l'intervallo per calcolare la media di durata va da 0 a 300 secondi

CallCenterSimulator.exe 2 25 0 250

Lancia il simulatore per valutare il modello basato su distribuzione normale inversa, con 25 agenti, e l'intervallo per calcolare la media di durata va da 0 a 250 secondi.

Al termine di ogni simulazione, a parte i dati riassuntivi più importanti, stampati sulla console, viene create anche un file di log con tutti i dettagli della simulazione.

5.4 Simulazioni e risultati

Le simulazioni sono state effettuate sfruttando i database di tre campagna pubblicitarie reali, gentilmente messi a disposizione dall'azienda Lascaux.
Le tre campagne sono le seguenti :

	Tot. Chia.	Risp.	Occup.	Non risp.	Fax	Errori	Silent Call	Agenti
Multi	492.508	46.758	22.636	184.038	3.573	9.938	225.565	n.d.
A	23.483	1.167	733	10.670	43	215	10.655	26
B	5.467	601	255	3.292	725	68	526	11
TOTALI	521.458	48.526	23.624	198.000	4.341	10.221	236.746	

(Tabella 1)

Osservando questi dati, si capisce subito che l'algoritmo predittivo usato realmente per queste campagne, non ha alcun controllo sull'effetto silent call.
Per le campagne Multi e A, quasi la metà delle chiamate, sono state delle silent call.
Evidentemente, il loro dialer, ha effettuato sempre un elevato numero di telefonate rispetto agli agenti a disposizione.
In seguito, il modello basato su ratio sarà riferito come Modello 0 (M0), il modello proposto, implementato con distribuzione normale, sarà riferito come Modello 1 (M1) mentre la versione basata su distribuzione normale inversa sarà riferito come Modello 2 (M2).

I test sono stati effettuati per tutti e tre i modelli secondo i seguenti criteri :

- a parità di campagna pubblicitaria, sono stati variati gli agenti, la ratio (valida solo per il modello 0), il limite superiore dell'intervallo per il calcolo della durata media di una chiamata
- sono stati presi in considerazione il numero di chiamate effettuate ed il tempo di conversazione totale, per tutti gli agenti, in una ora. Oltre sicuramente al numero totale di silent call calcolate dal simulatore.

Prima di analizzare i test nel loro insieme, è interessante capire, sia quanto influisca il fattore ratio per il modello 0, sia valutare l'importanza di scegliere un buon intervallo tra due differenti turni di chiamate.

Di seguito, i risultati delle simulazioni relative al modello 0, con 10 agenti, ratio variabile, intervallo 110s.
Silent Call = numero totale di silent call su tutte le ore lavorative
Tempo totale (sec.) = tempo totale di un'ora di conversazione di tutti gli agenti
Chiamate/h = numero di chiamate risposte dal cliente in un' ora

Modello 0 - Agenti: 10 - Ratio: 1,5 - Inter. : 110s			
	Silent Call	**Tempo tot. (s)**	**Chiam./h**
Multipla	38.547	40.601	946
A	3	2.827	94
B	16	2.873	32

Modello 0 - Agenti: 10 - Ratio: 1,7- Inter. : 110s			
	Silent Call	**Tempo tot. (s)**	**Chiam./h**
Multipla	38.636	40.430	967
A	3	2.668	92
B	20	2.618	28

Modello 0- Agenti: 10 - Ratio: 2 - Inter. : 110s			
	Silent Call	**Tempo tot. (s)**	**Chiam./h**
Multipla	38.596	40.341	930
A	4	3.133	97
B	30	2.220	23

Modello 0 - Agenti: 10 - Ratio: 3 - Inter. : 110s			
	Silent Call	**Tempo tot. (s)**	**Chiam./h**
Multipla	38.865	39.535	891
A	3	3.035	98
B	54	2.075	20

(Risultati 1)

Dai risultati si può intuire come l'aumento del fattore ratio influisca negativamente sul numero di silent call, che tendenzialmente aumenta. La principale conseguenza di questo fenomeno è una diminuzione delle chiamate utili, ovvero di minuti totali spesi in comunicazione con il cliente. Aumentando gli agenti, questo effetto sarebbe amplificato, il numero di silent call sarebbe ancora maggiore.

La spiegazione di questo comportamento è facilmente intuibile. Aumentando il fattore ratio e mantenendo costante il numero di agenti, vengono effettuate sempre più chiamate, ma gli operatori che possono rispondere sono sempre gli stessi.

Variando, invece, il limite superiore dell'intervallo, ecco cosa accade.

Modello 0, 10 agenti, ratio 2, intervallo variabile :

Modello 0 - Agenti: 10 - Ratio: 2 - Inter. : 30s			
	Silent Call	**Tempo tot. (s)**	**Chiam./h**
Multipla	41.050	25.442	602
A	6	2.914	93
B	34	2.220	23

Modello 0 - Agenti: 10 - Ratio: 2- Inter. : 80s			
	Silent Call	**Tempo tot. (s)**	**Chiam./h**
Multipla	39.221	35.678	863
A	4	3.138	98
B	30	2.292	23

Modello 0- Agenti: 10 - Ratio: 2 - Inter. : 130s			
	Silent Call	**Tempo tot. (s)**	**Chiam./h**
Multipla	38.362	41.459	935
A	4	3.104	96
B	29	2.312	23

Modello 0 - Agenti: 10 - Ratio: 2 - Inter. : 180s			
	Silent Call	**Tempo tot. (s)**	**Chiam./h**
Multipla	37.670	46.427	1064
A	4	3.006	94
B	28	2.332	23

(Risultati 2)

Modello 1, 10 agenti, intervallo variabile :

Modello 1 - Agenti: 10 - Inter. : 30s			
	Silent Call	**Tempo tot. (s)**	**Chiam./h**
Multipla	41.056	25.407	618
A	4	3.330	91
B	16	2.079	21

Modello 1 - Agenti: 10 - Inter. : 80s			
	Silent Call	**Tempo tot. (s)**	**Chiam./h**
Multipla	39.257	35.945	846
A	2	3.150	98
B	10	2.548	29

Modello 1 - Agenti: 10 - Inter. : 130s			
	Silent Call	**Tempo tot. (s)**	**Chiam./h**
Multipla	38.429	41.714	961
A	3	3.547	105
B	8	3.094	28

Modello 1 - Agenti: 10 - Inter. : 180s			
	Silent Call	**Tempo tot. (s)**	**Chiam./h**
Multipla	37.775	45.858	1.048
A	3	3.334	99
B	10	2.833	25

(Risultati 3)

Modello 2, 10 agenti, intervallo variabile :

Modello 2 - Agenti: 10 - Inter. : 30s			
	Silent Call	**Tempo tot. (s)**	**Chiam./h**
Multipla	41.076	25.421	610
A	5	3.653	95
B	14	2.101	21

Modello 2 - Agenti: 10 - Inter. : 80s			
	Silent Call	**Tempo tot. (s)**	**Chiam./h**
Multipla	39.049	35.699	845
A	2	2.830	90
B	12	2.337	27

Modello 2 - Agenti: 10 - Inter. : 130s			
	Silent Call	**Tempo tot. (s)**	**Chiam./h**
Multipla	38.096	41.858	971
A	2	2.573	90
B	6	2.167	23

Modello 2 - Agenti: 10 - Inter. : 180s			
	Silent Call	**Tempo tot. (s)**	**Chiam./h**
Multipla	36.419	45.600	1.034
A	2	2.871	96
B	7	3.000	29

(Risultati 4)

Osservando i risultati, si può notare, che per tutti e tre i modelli, più aumenta l'intervallo, e più ci sono telefonate utili, con un numero inferiore di silent call.

Si può capire, in base al modello, quale è l'intervallo migliore che porta a risultati più soddisfacenti.

Per quanto riguarda la campagna Multipla, l'intervallo [0,180] risulta il migliore per tutti i modelli. Essendo una campagna con numerose chiamate, più si amplia l'intervallo e più telefonate vengono prese in considerazione. Evidentemente, per questa campagna, l'intervallo potrebbe arrivare anche fino a 450, considerando il grafico (vedi capitolo 3, Grafico 8 e 9). Andando troppo oltre, invece, si avrebbe l'effetto contrario, una media troppo spostata verso le lunghe durate, che non porterebbe a buoni risultati.

Riguardo al modello 0, per la campagna A, l'intervallo migliore è [0,80]. Per la B, potrebbero andare bene sia 130 che 180, sono quasi equivalenti come scelta, in questo caso.

Per il modello 1,invece, è [0,130] la scelta migliore, sia per la campagna A che B.

Sul modello 2, non è possibile fare troppo considerazioni. Durante la predizione, utilizza sempre numeri diversi,ed ha, quindi, un comportamento che può variare di molto anche sulla stessa campagna.

Per capire quanto sia davvero importante l'intervallo scelto, si notino i risultati di un test, effettuato con la campagna Multipla, prima con 100 e poi con 50 agenti, un intervallo prima di 180 e poi di 400 secondi.

Agenti: 100 - Ratio: 2 - Inter. : 180s - C.Multi			
	Silent Call	**Tempo tot. (s)**	**Chiam./h**
M0	665	221.076	4.613
M1	267	216.887	4.517
M2	220	224.161	4.655

Agenti: 50 - Ratio: 2 - Inter. : 400s - C.Multi			
	Silent Call	**Tempo tot. (s)**	**Chiam./h**
M0	440	223.644	4.654
M1	116	218.840	4.534
M2	92	233.703	4.848

(Risultati 5)

Aumentando l'intervallo da 180 a 400 secondi, con la metà degli agenti si ottengono prestazioni uguali o migliori.
Anche in questo caso, la spiegazione del fenomeno è semplice. E' inutile effettuare nuove chiamate ogni K secondi, se la durata media della maggior parte delle telefonate, è spostata bel oltre il valore K.

Le differenze tra i modelli, non sono molto evidenti, quando la simulazione avviene con solo 10 agenti. Si può notare una tendenza del modello 2 ad avere meno silent call, ma anche meno chiamate in un'ora, rispetto agli altri due modelli considerati.
Per cercare di capire, se, e quanto, il modello 2 può essere migliore degli altri due, bisogna utilizzare più agenti.
I seguenti risultati, riguardano i tre modelli, con la campagna Multipla, numero agenti variabile, intervallo 180 e ratio 2.

Agenti: 30 - Ratio: 2 - Inter. : 180s - C.Multi			
	Silent Call	**Tempo tot. (s)**	**Chiam./h**
M0	21.121	127.585	2.649
M1	21.150	125.920	2.592
M2	20.886	127.760	2.648

Agenti: 50 - Ratio: 2 - Inter. : 180s - C.Multi			
	Silent Call	**Tempo tot. (s)**	**Chiam./h**
M0	8.900	176.246	3.657
M1	7.883	182.632	3.770
M2	7.724	185.389	3.816

Agenti: 100 - Ratio: 2 - Inter. : 180s - C.Multi			
	Silent Call	**Tempo tot. (s)**	**Chiam./h**
M0	665	221.076	4.613
M1	267	216.887	4.517
M2	220	224.161	4.655

(Risultati 6)

Con la suddetta configurazione, è visibile una tangibile differenza tra i tre modelli.
Sicuramente quello più performante è il modello 2, che garantisce un numero minore di silent call, un numero di chiamate superiore, per ogni ora di lavoro, e quindi secondi in più di conversazione con il cliente.
Il modello 0 ed 1, si equivalgono quasi, se non fosse per il numero elevato di silent call che contraddistingue il modello basato su ratio.
Sulla base dei precedenti risultati, si può riscontrare che il modello 2 genera un numero di silent call inferiore del 27% rispetto al modello 0 e del 7% rispetto all'uno.
I secondi in più di conversazione sono in entrambi i casi poco più del 2% rispetto agli altri due modelli.

Alla luce di questi risultati, si può affermare che il modello basato su ratio ha il forte vantaggio di non necessitare di conoscenza storica per poter funzionare.
Non ha bisogno di effettuare elaborazione di dati di precedenti campagna pubblicitarie.
E' semplice da utilizzare e da gestire, garantisce un elevato numero di secondi di conversazione. Ma non è la scelta corretta, per tutti quei Call Center che si trovano in paesi ove vigono leggi restrittive sul numero massimo di silent call al giorno.
Il modello basato su distribuzione normale, soffre di una importante perdita di precisione, dovuta al fatto che prende in considerazione solo un ristretto intervallo di durata. Questo, porta a predizioni errate, che si tramutano in un alto numero di silent call ed un tempo minore speso in conversazione con il cliente.
Necessità di conoscenza storica, ed è quindi più costoso, sia in termini di denaro che di tempo, rispetto al modello basato su ratio. I vantaggi rispetto a quest'ultimo, sono minimi. Per i Call Center che non hanno alcuna limitazione sulle silent call, converrebbe più il modello 0 che 1.
Il modello basato su distribuzione normale inversa, sembra il migliore, a giudicare dai risultati. Con pochi agenti a disposizione, però, non è così tanto netta la sua convenienze rispetto ai modelli 0 ed 1. Quando il numero di agenti è superiore alle poche decine, questo modello riesce a fare la differenza, in termini di chiamate utili in una ora, ma soprattutto, fa la differenza riguardo al fattore silent call.

E' più preciso del modello 1 perché tiene in considerazione tutte le chiamate, non solo quelle che hanno una certa durata massima. Anche questo modello, però, necessita di elaborare dati riguardanti precedenti campagne prima di poter essere usato.
E' logico pensare che maggiori sono i dati a disposizione da elaborare, e più precisa dovebbe essere la predizione del numero di chiamate da effettuare.

Riguardo al simulatore, la sua validità è dimostrata dal fatto che gli andamenti dei tre modelli hanno rispettato le aspettative della teoria (vedi capitolo 3).
Non è sicuramente molto preciso, poiché, nei test con 10 agenti, ad esempio, ha fatto registrare anche 45.600 secondi di conversazione in una sola ora. Questo vorrebbe dire 760 minuti, che diviso il numero di agenti sarebbe 76 minuti in una ora.
E' evidente che l'orologio interno del simulatore è più veloce rispetto al reale scorrere del tempo. Ma è un errore accetabile, se si pensa che è cumulativo su quasi mezzo milione di record.

Conclusioni

L'intero lavoro può essere diviso in tre macro sezioni: individuazione e studio del problema, proposta di una soluzione e confronto con quelle già presenti ed, infine, implementazione e test della nuova soluzione introdotta.

Durante la prima fase, c'è stata una approfondita ricerca su come funzionano i Call Center, come sono nati e come sono attualmente. Quali sono stati i motivi per cui oggi hanno assunto una particolare conformazione strutturale e quali sono i principali problemi che devono affrontare. Sicuramente, il Predictive Dialing è stata una efficace soluzione che ha ridotto, quasi completamente, i tempi "morti" di un operatore. Le silent call, però, hanno messo in evidenza un effetto collaterale di tale tecnologia. Un cattivo utilizzo di algoritmi predittivi può portare a conseguenze negative sia per il Call Center stesso che per i clienti. La poca letteratura a disposizione, ha evidenziato una idea, ormai obsoleta, di Predictive Dialing, limitata anche dalla situazione tecnologica di quindici anni fa. Il modello di Samuelson è sicuramente ancora attuale sotto alcuni aspetti. Gli indicatori di valutazione che lui aveva individuato, si sono rivelati utili e validi anche oggi. Attualmente, però, sarebbe sbagliato usare un modello che è stato costruito su due fattori, linee ed agenti, che anni fa erano un limite, ma ora non lo sono più. Per questo, il modello di Samuelson è stato superato, ma può servire come spunto o punto di partenza per nuove soluzioni. Il modello basato su "ratio" è completamente inadatto per i sistemi moderni, garantisce un buon margine di minuti di conversazione al prezzo di un numero elevatissimo di silent call.

Per questo motivo, nella seconda parte del lavoro, è stata introdotta una soluzione che si adatta alle nuove tecnologie, ed è conforme anche alle attuali normative in vigore. Studiando gli esiti di oltre mezzo milione di telefonate, di diverse campagne pubblicitarie, è risultato chiaro l'andamento delle durate delle stesse. Quasi tutte, terminano entro i primi 2 o 3 minuti. Per questo motivo, in un primo momento, è stato pensato di utilizzare la distribuzione normale per approssimare tale fenomeno.

Limitando, però, lo studio ad un intervallo di durata abbastanza ristretto. Successivamente, considerando che le telefonate escluse dalla predizione erano, in media, il 30%, è stato pensato di sfruttare la distribuzione normale inversa, per non avere una perdita di informazione così importante. Quest'ultima, ha una caratteristica

fondamentale, approssima fenomeni che hanno una elevata quantità di valori approssimabili con la distribuzione normale, ma che hanno anche altri valori, che vanno oltre la campana di Gauss. Questa seconda soluzione ha permesso di prendere in considerazione tutte le possibili durate, e non solo quelle relative ad un certo intervallo.
Nell'ultima parte del lavoro, il modello proposto è stato implementato con il linguaggio C#.Net/Mono, ma in realtà può essere utilizzato un qualsiasi altro linguaggio.
Inoltre, il modello è completamente indipendente dal server PBX, sottostante. La sua logica può essere utilizzata anche con altri sistemi.
I risultati dei test, hanno confermato ciò che la teoria aveva anticipato. Il modello che approssima il fenomeno con la distribuzione normale inversa, è sicuramente più preciso e più efficiente di quello che si basa sulla distribuzione normale. Le silent call sono un problema costante del modello basato su ratio e sono troppe anche per il modello basato su distribuzione normale. Segno evidente, che le predizioni limitate ad un intervallo ristretto, sono imprecise e portano ad alto numero di chiamate inutili.
Attualmente la migliore soluzione sarebbe un server PBX come Asterisk, utilizzato come smart-dialer. Un sistema che, ad un costo di acquisto nullo, garantisce una elevatissima efficienza, poiché, i clienti che arrivano a parlare con gli operatori, sono realmente interessati alle offerte. Grazie alla voce interattiva, uno smart-dialer informa ogni cliente del motivo per cui è stato contattato, lasciando libera scelta di decidere di proseguire o chiudere la conversazione.
Il futuro, in parte anche già il presente, vede come mezzo di comunicazione dei Call Center, non solo le linee telefoniche PSTN o VoIP, ma anche sms ed email. Oltre al fatto di sfruttare database di numeri, non più presi a caso, ma selezionati in base a delle regole ben precise, stabilite sulla base delle preferenze e dei gusti di ogni utente.
Asterisk si è rivelata una piattaforma completa, molto potente e sicuramente stabile.
Durante tutti i mesi di utilizzo non si è mai verificato un errore dovuto al server.
Tutti i micro problemi incontrati con Asterisk sono stati sempre risolti velocemente grazie alla documentazione presente in rete. Sicuramente una ottima scelta per tutte le aziende, piccole o grandi che siano. Senza particolari modifiche, Asterisk è pronto per essere utilizzato come centralino e per servire da inbound Call Center per il supporto telefonico. Se dovesse essere utilizzato come outbound Call Center, sarebbe necessario

collegarlo ad un modulo esterno per il Predictive Dialing, proprio come è stato fatto durante questo lavoro.

Lo svolgimento di questo lavoro di tesi, mi ha permesso di entrare in contatto con un mondo, a me, del tutto sconosciuto fino a sei mesi fa.

L'idea diffusa di Call Center è quella di un luogo dove tante persone sono sedute avanti ad un terminale, munite di microfono e cuffia.

Grazie a questa esperienza, ho potuto constatare, di persona, che in realtà un Call Center, sia inbound che outbound, è un sistema molto complesso, che funziona grazie all'intergrazione, quasi perfetta, di tante tecnologie differenti tra loro.

E' un interessante laboratorio per sistemi informatici innovativi, che facilitano sempre di più il compito degli operatori. In alcuni casi, come l'IVR (Interactive Voice Response), il sistema automatizzato si sostituisce interamente al lavoro dell'agente.

La parte più complicata di tutta la tesi è stata certamente l'individuazione dell'attuale letteratura in merito. In quindici anni, è stato scritto un solo vero articolo, il cui argomento era il Predictive Dialing, quello di Samuelson.

Certamente, i pochi controlli sulle attività dei sistemi di telemarketing, non hanno stimolato le aziende a cercare nuove soluzioni, che limitassero l'effetto indesiderato delle silent call. Tutti hanno preferito seguire la strada più facile e meno costosa, inondare la rete telefonica pubblica di chiamate verso i numeri di tutto il mondo.

Come sempre, però, l'esagerazione, prima o poi, si scontra con la pazienza di chi non è più disposto a sopportare. Per questo motivo, ad oggi, tutti i Call Center americani ed inglesi, e nei prossimi mesi, anche quelli italiani, dovranno affrontare il problema 'silent call'. Considerando questo nuovo aspetto, il Predictive Dialing non deve essere visto, solamente, come una tecnica per massimizzare le vendite, ma anche come un elemento di controllo e di ottimizzazione nel rispetto delle leggi e dei clienti.

Cimentarmi nell'individuazione di un modello risolutivo per questo problema, è stato sicuramente non facile, ma altamente formativo. E' stato un lavoro lungo ma, anche, divertente. Che mi ha regalato qualche soddisfazione, quando, tramite il simulatore, i risultati hanno dato ragione alla mia idea. Indipendentemente dai modelli usati, il Predictive Dialing rimane sicuramente un fondamentale strumento di ottimizzazione di un outbound Call Center, da cui possono dipendere i successi o gli insuccessi del telemarketing. Nonostante siano quindici anni, che questa tecnologia è stata introdotta,

rimane un problema irrisolto, in cui, tante aziende si sono cimentate, ognuna con risultati più o meno soddisfacenti. Ancora nessuno, però, è riuscito a far lavorare un agente per un'ora intera, senza pause. Poter lavorare in questo campo di ricerca è stata una esperienza entusiasmante e sicuramente stimolante. Essendo un settore dove ancora nessuno ha inventato "La Soluzione", la speranza che il proprio modello possa essere la risposta al problema, è sempre viva.

Bibliografia

1. Call Center, gli schiavi elettronici della new economy - Novembre 2005 - Claudio Cugusi - ISBN 88-7563-149-2

2. An Overview of Routing and Staffing Algorithms in Multi-Skill Customer Contact Centers – Marzo 2006 - Ger Koole & Auke Pot - Department of Mathematics, Vrije Universiteit Amsterdam, The Netherlands

3. Markov chain models of a telephone call center with call blending – Alexandre Deslauriers,Pierre L'Ecuyer, Juta Pichitlamken,Armann Ingolfsson, Athanassios N.Avramidis - D´epartement d'Informatique et de Recherche Op´erationnelle Universit´e de Montr´eal, C.P. 6128, Succ. Centre-Ville Montr´eal, H3C 3J7, CANADA

4. PBX Systems for IP Telephony - Cap. 1 – Aprile 2002- Allan Sulkin – ISBN 0071375686

5. Asterisk: The Future of Telephony - O'Reilly, 2007 - Jim Van Meggelen, Leif Madsen, Jared Smith - ISBN 0596510489, 9780596510480

6. Altesys SpA, Via Verdi 20 - 20090 Assago (MI) - www.altesys.com/doc/Borsani2007/Ottobre07/Comunicati%20stampa/Comunicato%20stampa%20ProDialer_Altesys.pdf

7. Using simulation to predict market behavior for outbound call centers - 2007 - Paulo J. de Freitas Filho, Geovani Ferreira da Cruz, Rui Seara, Guilherme Steinmann - http://portal.acm.org/citation.cfm?id=1351940

8. Predictive Dialing for Outbound Telephone Call Centers – INTERFACES Vol. 29, No. 5, September-October 1999, pp. 66-81- Douglas A. Samuelson

9. Telephone Call Centers: Tutorial, Review, and Research Prospects - Manufacturing & Service Operations Management 5:79–141, 2003 – Noah Gans, Ger Koole, Avishai Mandelbaum

10. Queueing Models of Call Centers: An Introduction - Annals of Operations Research 113 (2002) 41- 59 – Ger Koole, Avishai Mandelbaum

11. Call Center Mathematics - A scientific method for understanding and improving contact centers - Ger Koole

12. Information Theory And The Central Limit Theorem – Luglio 2004 - Oliver Johnson (University of Cambridge, UK) - ISBN 978-1-86094-473-4

13. Normal Distribution, Characterizations with applications - Lecture Notes in Statistics 1995, Vol 100- Wlodzimierz Bryc, Department of Mathematics, University of Cincinnati

14. The normal inverse Gaussian distribution: a versatile model forheavy-tailed stochastic processes - Proceedings. (ICASSP apos;01). 2001 IEEE International Conference onVolume 6, Issue , 2001 vol.6 - Hanssen, A., Oigard, T.A.

Indice delle figure

Indice dei grafici

Indice delle tabelle

Ringraziamenti

Sentiti ringraziamenti vanno al Professor Antonio Cisternino ed a Fabio Testa della Lascaux, che sono stati sempre gentili e disponibili, soprattutto nei momenti di maggiore difficoltà.

Un ringraziamento speciale anche a Carmen, una persona per me molto importante, che con il suo sorriso e la sua dolcezza è sempre riuscita a trasmettermi la calma di cui avevo bisogno. Grazie!

Come non ringraziare anche il mio amico Carlo, che da anni mi insegna a vedere sempre il lato positivo delle cose. Caro Licx, alla fine ci siamo riusciti!
Quante sere mi hai visto distrutto avanti il monitor aspettando l'ispirazione!?

Un grazie va anche a tutti gli amici, specialmente quelli di sempre, che con semplici domande mi hanno fatto capire che quello che stavo facendo era importante...anche per loro. Grazie uagliò!